

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

**SIMPLE QUORUM DRIVE FOR THE FreeBSD
CTL HA AND THE BEAST STORAGE SYSTEM**

**CONTROL YOUR FILES USING YOUR
OWN CLOUD WITH OWNCLOUD**

INTERVIEW WITH ARYEH FRIEDMAN

**OPENBSD 6.0 SECURITY FEATURE
CHANGES AND PLEDGE SECURITY INTERFACE**

THE STATE OF MULTIMEDIA IN OPENBSD

**BSD:
IT REALLY NEVER DIES**

**VOL 10 NO 13
ISSUE 12/2016 (89)
1898-9144**

Dear Readers,

I hope you have all been well, safe and warm. We have been having a crazy and unexpected winter attack in Europe. It's been great enjoying all the snow like how I used to when I was a kid. But in some regions it has been pretty dangerous, so I hope you were all prepared.

Crazy winter weather is almost finished as well as my adventure with BSD Magazine. I would love to thank all of you for being with me during my work as Product Manager in the Mag. Thank you for your support, kind words, help and feedback.

We have been carrying on long disputes, whether we should close the Mag or keep the project. Just now, as I'm writing, I've received a final decision about Ewa, the previous Product Manager, taking it back. With this positive information, let's hop right into the issue.

The issue is being opened by Mikahail Zakharow's article, Simple Quorum Drive for the FreeBSD CTL HA and the BeaST Storage System. Staying in FreeBSD Corner, you will read Pedro Giffuni's article BSD:It never really dies.

We have two great articles about OpenBSD this time around. One is from our all time favorite author, David Carlier, who shared with us his thoughts in The State of Multimedia in OpenBSD. From thesecurity standpoint, Albert Hui shared his thoughts on OpenBSD 6.0 Security Feature Changes and Pledge Security Interface.

And for all of you who have been asking about own-Cloud, here it is! Marcus Schmitt shows us how to Control Your Files Using Your Own Cloud With own-Cloud.

Next, you will read an interview with Aryeh Friedman, CTO and co-CEO of Friedman-Nixon-Wong Enterprises, LLC.

Last, but not least, Rob Somerville contributes his insight on failure in communication and toxic people.

Enjoy reading! I wish you all the best of 2017, and continued satisfaction from upcoming BSD Mag issues.

Marta & BSD Team

MAGAZINE **BSD**

Editor in Chief:

Marta Ziemianowicz

marta.ziemianowicz@software.com.pl

Contributing:

Mikhail Zakharov, Pedro Giffuni, David Carlier, Albert Hui, Marcus Schmitt, Aryeh Friedman and Rob Somerville.

Top Betatesters & Proofreaders:

Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omo-kanwaye, Radjis Mahangoe, Katherine Dizon and Mark VonFange.

Special Thanks:

Annie Zhang

Denise Ebery

DTP:

Marta Ziemianowicz

Senior Consultant/Publisher:

Paweł Marciniak

pawel@software.com.pl

CEO:

Joanna Kretowicz

joanna.kretowicz@software.com.pl

Publisher:

Hakin9 Media SK 02-676 Warsaw, Poland Postepu 17D Poland worldwide
publishing editors@bsdmag.org www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail:
editors@bsdmag.org.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

CONTENTS

News

[BSD World Monthly News](#) **5**

by Marta Ziemianowicz

This column presents the latest news coverage of events, product releases and trending topics.

FreeBSD Corner

[Simple Quorum Drive for the FreeBSD CTL HA and the BeaST Storage System](#) **16**

by Mikhail Zakharov

During our experiments on developing the BeaST storage system we faced the lack of an automatic LUN failover function of the CTL HA subsystem. Yes, we can switch LUNs with CTL HA, but we have to do it manually setting “Primary role” to the alive controller: `sysctl kern.camctl.ha_role=0`

We also have to do it fast enough, otherwise, a client host may lose access to the drives of the storage system.

[BSD: It Really Never Dies](#) **30**

by Pedro Giffuni

This year, I was one of three developers representing FreeBSD at the Google Summer of Code 2016 Mentor Summit in Sunnyvale, California. It was a nice opportunity to get in touch with other projects and let FreeBSD get a little better known. We don't really have many chances to meet other developers from our own project so during the dinner, we started discussing a variety of matters. While we knew there was someone from NetBSD, we hadn't seen anyone from the other BSD projects. One of my colleagues made a comment along the lines of “perhaps NetBSD should be reconsidering its mere existence,” to which I replied ... “that will never happen, pretty much as we will not ever consider closing our project just because another project is more popular.”

OpenBSD

[The State of Multimedia in OpenBSD](#) **35**

by David Carlier

OpenBSD is often seen as an excellent operating system for routers, in which it indeed excels.

In this article, we will see that OpenBSD can have more use cases, thus being used for sound editing, video games or multimedia at large.

[OpenBSD 6.0 Security Feature Changes and Pledge Security Interface](#) **42**

by Albert Hui

The OpenBSD operating system is well known to be security focused. For the 6.0 release, a noteworthy change is that systrace, support for Linux emulation and the usermount option has been removed. If interested, I highly recommend you watch Theo de Raddt's excellent talk at dotSecurity 2016.

ownCloud

[Control Your Files Using Your Own Cloud With ownCloud](#) **45**

by Marcus Schmitt

If you search for cloud solutions within the internet, you are able to find hundreds of solution providers. But what happens with your private data? Who really has access to your data? Very restrictive file access settings within the cloud service doesn't guarantee that the cloud company won't copy or forward your private files. Furthermore, some providers simply scan your data to create a user profile in order to sell the profile to marketing organisations.

Interview

[Interview with Aryeh Friedman](#) **57**

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

Rob's Column

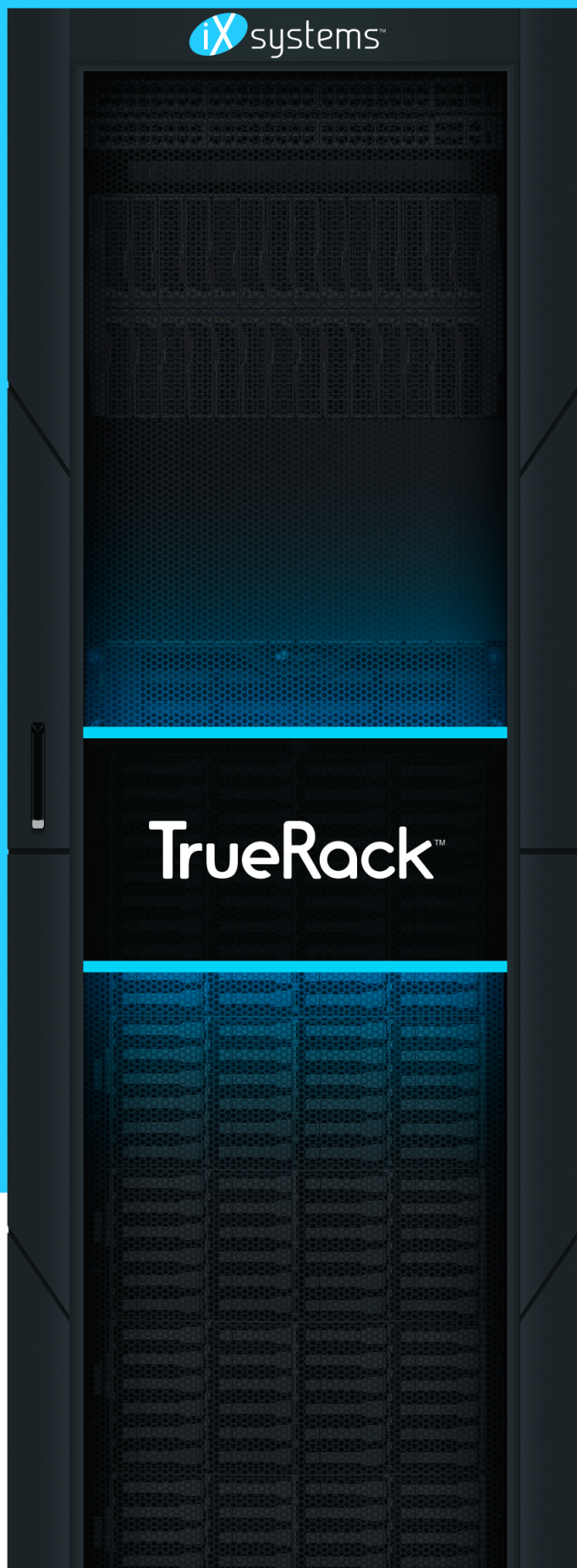
71

by Rob Somerville

More than 30 years have passed since he first typed at a computer keyboard. Rob Somerville looks back over the major changes in the IT industry and enquires what will be the next revolution on the horizon.

Simplify your Data Center

Meet TrueRack™ — A powerfully flexible rack-scale architecture that takes the guesswork out of building large scale data center applications.



- Converged Infrastructure
- Customizable for Virtualization, Big Data, Cloud & Hyperscale
- Up to 70% Lower TCO Than AWS
- Scalable and Repeatable Deployments

For more information on TrueRack, visit **ixsystems.com/TrueRack** today

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BSDP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BSDP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BSDP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

Lumina 1.2 Desktop Environment Released

A new release of Lumina is now available to ring in 2017, the BSD-first Qt-powered open-source desktop environment.

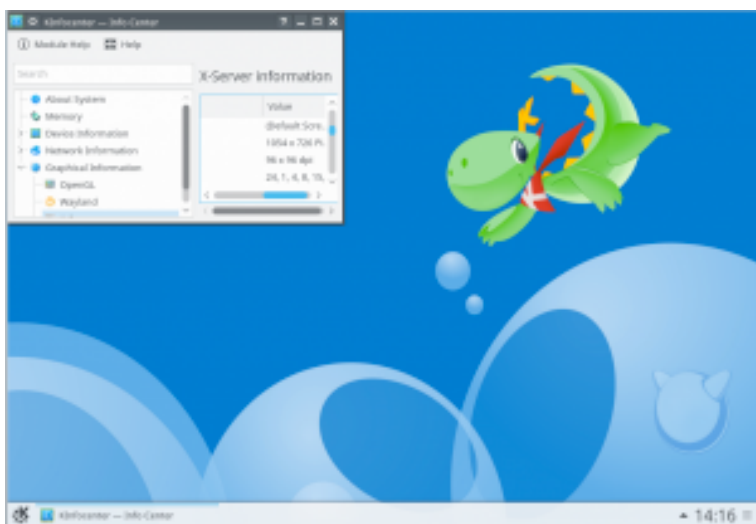
With today's Lumina 1.2 desktop environment, the libLuminaUtils.so library is no longer used/needed, the internal Lumina Theme engine has been separated from all utilities, there are new panel and menu plug-ins and a new Lumina Archiver utility as a Qt5 front to Tar. The new plug-ins are an audio player, JSON menu, and a lock desktop menu plugin for locking the current session.

Great to see the low-level code improvements for Lumina 1.2 in reducing the library dependencies plus a wide range of other minor changes to the desktop itself and its utilities. Lumina continues to be the only desktop option to be offered by the FreeBSD-based TrueOS while it continues to be packaged for various BSDs and Linux distributions.

Those wishing to learn more about Lumina 1.2.0 can do so at Lumina-Desktop.org.

http://www.phoronix.com/scan.php?page=news_item&px=Lumina-1.2-Released

KDE Frameworks 5 & KDE Plasma 5 now on FreeBSD



Developer adridg has announced that KDE Frameworks 5 and KDE Plasma 5 Desktop environment is now available on FreeBSD, with support for Wayland still in development.

It's been quiet from the KDE-FreeBSD folks for a bit, but not because it's actually been quiet. Tobias has been on a roll, and Dima has started doing stuff again, and Gleb is still watching over some ports, and Raphael is hovering over it all with good advice. So

here's some bits and pieces:

Some time ago I mentioned a branded wallpaper for FreeBSD, based off of the Flying Konqui wallpaper — which in turn I had mentioned in February. Anyway, here's a screenshot of the if-it's-up-to-me default wallpaper for Plasma 5 on FreeBSD. It's running in VirtualBox, which is why KInfoCenter reports an interesting resolution (KInfoCenter has also been expanded with a lot better data on FreeBSD hosts, so that it reports sensible memory use, and sensible disk usage).

Original article: <http://euroquis.nl/bobulate/?p=1521>

Related:

<http://news.softpedia.com/news/kde-frameworks-5-and-kde-plasma-5-desktop-landed-on-freebsd-wayland-coming-soon-511400.shtml>

<https://www.freebsdnews.com/2017/01/06/kde-frameworks-5-kde-plasma-5-desktop-landed-freebsd/>

The FreeBSD 64-bit Base System Can Now Be Linked Using LLD

LLVM's LLD linker has been making a lot of progress over the past year and now it's hit the milestone of being able to link the entire FreeBSD/amd64 base system.

FreeBSD developers have been looking to use LLD as their system linker, following their move a while back now of using LLVM/Clang in place of GCC.

As of last week, one of the developers working on this project was able to link the entire FreeBSD/amd64 base system with user-space world and kernel using LLD.

Perhaps by FreeBSD 12.0 we'll see them migrate to the LLD linker by default. As covered in the earlier article about LLD's progress, this LLVM linker remains much faster than GNU Gold and alternatives. The code-base is arguably cleaner, more modern, and other benefits.

http://www.phoronix.com/scan.php?page=news_item&px=FreeBSD-LLD-Success

FreeBSD Foundation 2016 Year End Recap



The FreeBSD Foundation has released a recap of the month of December, as well as extend their thanks to the greater FreeBSD community, including YOU. View the links below for a summary of development projects, fundraising initiatives, conferences, advocacy efforts, and more.

Dear FreeBSD Community Member,

Wow, this has been quite the year! As we reflect back on 2016, we are proud of the support we've been able to provide to the FreeBSD Project. This includes providing more operating system improvements, leading the FreeBSD 11 release efforts, providing more FreeBSD advocacy and education support, improving the FreeBSD infrastructure, supporting more face-to-face opportunities, and promoting FreeBSD in new regions, like China and India. In this newsletter you'll find many articles talking about some of these areas we've supported. Please, take a minute to read what the Foundation has done this past year to support FreeBSD. If you haven't made a donation to the Foundation yet, please consider making one today!

Deb

FreeBSD Foundation December 2016 Update:

<https://www.freebsd.foundation.org/wp-content/uploads/2016/12/FreeBSD-Foundation-December-2016-Update.pdf>

Recap of 2016 Advocacy Efforts:

<https://www.freebsd.foundation.org/blog/recap-of-2016-advocacy-efforts/>

<https://www.freebsdnews.com/2017/01/06/freebsd-foundation-december-2016-update/>

Haiku OS Gaining Ground On UEFI, FreeBSD Compatibility Layer, Remote Debugging

For those interested in the BeOS-inspired Haiku open-source operating system, they have issued their latest monthly progress report to end out 2016.

Last time we were talking about Haiku OS on Phoronix it was about working on UEFI support and that continued throughout December. Haiku OS now has a simple frame-buffer driver for UEFI, some key fixes have landed, serial debug support was added to investigate issues when booting with UEFI, improved GPT partition creation, and other improvements.

December also brought improvements to Haiku OS when it comes to their launch daemon, x86_64 packages, their remote debugging interface, kernel/driver fixes, BMediaClient API, and more. Those interested in learning more about Haiku's latest development work can read this status update.

http://www.phoronix.com/scan.php?page=news_item&px=BeOS-Haiku-December-Ends-2016

DragonFlyBSD Working On NUMA-Awareness, Memory Changes

Matthew Dillon's latest work on the DragonFlyBSD kernel includes steps towards supporting NUMA-awareness, locking, and other memory allocation related changes.

Dillon has been making a number of changes related to DragonFlyBSD's memory handling as well as locking within the kernel. These changes are helping DragonFly's performance on multi-socket systems. He shared, "As an example of what we get from this, the dual Xeon system was topping out at 1.5-2M zero-fill page faults a second running a test program on all 32 threads before the changes. After the changes (and all the other work), the same system is now pushing 5.6 MILLION zero-fill page faults/sec across 32 threads. Our four-socket Opteron system also saw major improvements and can now achieve something like 4.7M zero-fill page faults a second across the 48-cores."

The NUMA-awareness for DragonFlyBSD is also further increasing performance and reducing memory stalls as well. If you want to learn more about these changes going into DragonFlyBSD, read this mailing list post for all the details, as well as some more performance numbers.

http://www.phoronix.com/scan.php?page=news_item&px=DragonFlyBSD-NUMA-Memory-Work

Faster Raspberry Pi X.Org Desktop Performance With NEON

Broadcom developer Eric Anholt has begun writing code within the VC4 open-source driver stack to make use of NEON in its acceleration code-paths.

NEON, of course, being ARM's Advanced SIMD extension. With making use of NEON, the VC4 driver is seeing faster X performance.

Eric commented on some of the performance impact:

"It seems to be intended for stack loads/stores, but we can also use it to get 64 bytes of data in from memory untouched into NEON registers, and then I can use 4 (32bpp) or 8 (8 or 16bpp) VST1s to store it to the CPU side. With this, we get a 208.256% +/- 7.07029% (n=10) improvement to GetTexImage performance at 1024x1024. Doing the same NEON code for stores gave a 41.2371% +/- 3.52799% (n=10) improvement, probably mostly due to not calling into memcpy and having it go through its size/alignment-based memcpy path choosing process.

I'm not yet hitting full memory bandwidth, but this should be a noticeable improvement to X, and it'll probably help my piglit test suite runtime as well."

Those interested in learning more can read this week's VC4 status update where he talks about the latest work on improving the speed of this graphics driver for the Raspberry Pi.

http://www.phoronix.com/scan.php?page=news_item&px=Raspberry-Pi-VC4-NEON-Gains

NetBSD 7.1 RC1 Released

The first release candidate of the upcoming NetBSD 7.1 is now available for testing.

NetBSD 7.1 has been queuing up with many kernel changes, an updated snapshot of xf86-video-nouveau, improved support for Google Compute Engine, and a variety of other changes outlined here.

Download links and other details on NetBSD 7.1 RC1 via NetBSD.org.

http://www.phoronix.com/scan.php?page=news_item&px=NetBSD-7.1-RC1-Released

FabioLolix/BSD-Timeline

Fabio shared with us his BSD Timeline. Take a look and contribute!

17.01 (2017-01-15)

- Total: 24
- Added: MidnightBSD, FreeNAS, NAS4Free, pfSense, TrueNAS, Fugulta
- Added: BSD Router Project, SmallWall, t1n1wall, RaspBSD, MirOS
- Added: OpenNAS, HardnedBSD

v16.12

FabioLolix released this 25 days ago · 28 commits to master since this release

16.12 (2016-12-25)

- Total: 11
- Added: GhostBSD
- Added: release versions for NetBSD, OpenBSD, FreeBSD, DragonflyBSD, PC-BSD
- Renamed: PC-BSD to TrueOS
- Discontinued: m0n0wall, DesktopBSD
- Fixed: DesktopBSD start date
- Works best with gnuclad 0.2.4
- When using LibreOffice Calc, turn on "quoted field as text"

<https://github.com/FabioLolix/BSD-Timeline>

Raspberry Pi Compute Module 3 Launched

The Raspberry Pi Foundation this morning announced the Compute Module 3 (CM3) as the successor to their original Compute Module.

The Raspberry Pi Compute Module remains targeted as an offering for those manufacturing customized products based on the Raspberry Pi. The Compute Module uses a DDR2 SO-DIMM interface and makes it easy and low-cost to integrate within custom hardware designs.

With the Compute Module 3, they've upgraded it to offering the same SoC and capabilities as the Raspberry Pi 3, which is much faster than the CM1 matching the hardware specs of the original Raspberry Pi 1. This means twice the amount of RAM and roughly 10x better CPU performance.



The Compute Module 3 has the BCM2837 SoC, 1GB RAM, and 4GB of eMMC memory. There is also now a Compute Module 3 Lite (CM3L) that has the BCM2837 and 1GB of RAM while the storage must be wired in separately via eMMC/SD. The CM3 will be priced at \$30 USD while the CM3L is priced at \$25.

Those interested in more information on the Raspberry Pi Compute Module 3 hardware can find out all of the details via this announcement.

http://www.phoronix.com/scan.php?page=news_item&px=RPi-Compute-Module-3

FreeBSD Security Advisory: OpenSSH vulnerabilities 1/11/2017

A FreeBSD Security Advisory concerning an OpenSSH vulnerability has recently been issued. You can view the full description of the vulnerability and solution on the mailing list page. User Vivek Gite also provides a solution for the problem below.

I. Background

OpenSSH is an implementation of the SSH protocol suite, providing an encrypted and authenticated transport for a variety of services, including remote shell access. OpenSSH supports accessing keys provided by a PKCS#11 token.

II. Problem Description

The ssh-agent(1) agent supports loading a PKCS#11 module from outside a trusted whitelist. An attacker can request loading of a PKCS#11 module across forwarded agent-socket. [CVE-2016-10009].

When privilege separation is disabled, forwarded Unix domain sockets would be created by sshd(8) with the privileges of 'root' instead of the authenticated user. [CVE-2016-10010].

Original announcement:

<https://lists.freebsd.org/pipermail/freebsd-security-notifications/2017-January/000305.html>

OpenSSH is critical for both sysadmin and programmers. It is an implementation of the SSH protocol suite, from OpenBSD project. It provides an encrypted session to your server.

OpenSSH multiple vulnerabilities

OpenSSH has multiple vulnerabilities as of 11th January 2017 running on FreeBSD operating system. From the advisory:

The ssh-agent(1) agent supports loading a PKCS#11 module from outside a trusted whitelist. An attacker can request loading of a PKCS#11 module across forwarded agent-socket. [CVE-2016-10009]

When privilege separation is disabled, forwarded Unix domain sockets would be created by sshd(8) with the privileges of 'root' instead of the authenticated user. [CVE-2016-10010]

Patch your FreeBSD server:

<https://www.nixcraft.com/patch-your-freebsd-server-for-openssh-vulnerabilities-11jan2017/168/>

<https://www.freebsdnews.com/2017/01/13/patch-freebsd-server-openssh-vulnerabilities-11jan2017/>

Red Hat's OpenShift Container Platform openly shifts storage into the hands of devs



Dynamically allocate without an admin over your shoulder

Enterprise Linux biz Red Hat has revised its OpenShift Container Platform to include support for dynamic storage provisioning in local and remote applications.

The software is an on-premises platform-as-a-service product that allows organizations to run applications using Kubernetes orchestration and Docker containers.

The latest iteration of the software, OpenShift Container Platform 3.4, provides on-the-fly container-native storage through Red Hat Gluster Storage, a software-defined file storage system for on-premises and public cloud installations.

The software now allows developers to allocate storage as needed and to deploy it with minimal effort.

In a statement, Ashesh Badani, VP and general manager of OpenShift at Red Hat, said the update addresses "the growing storage needs of both stateful and stateless applications across the hybrid cloud, allowing for coexistence of modern and future-forward workloads on a single, enterprise-ready platform."

Joe Fernandes, senior product manager for OpenShift, in a phone interview with The Register, explained that OpenShift Container Platform has supported stateful applications and storage since the company transitioned to its software to support Kubernetes and Docker a year and a half ago.

Previously, said Fernandes, adding storage required the involvement of an administrator. "Dynamic provisioning means being able to spin up the storage for each application when the developer configures it in real time," he said.

The update also improves multi-tenancy capabilities, made possible through Kubernetes namespaces, which subdivide clusters. Development teams can now separately search for project details and manage project membership through a revised web console.

What's new, said Fernandes, is the multi-tenant management through OpenShift. "It eases the burden on the administrator who is configuring the system for an organization," said Fernandes. "For our largest customers, they may have hundreds of tenants on the platform. They don't want to set up different clusters for each group."

Customers will also have access to reference guides for running the software on different infrastructure, including Amazon Web Services, Google Cloud Engine, Microsoft Azure, and OpenStack.

"What we want to do is provide more details on the best way to manage and install the software on different providers," said Fernandes.

Fernandes said Red Hat is working on implementing Kubernetes federation, to simplify the management of multiple clusters. When the API is more mature, he anticipates it will appear in a future update.

OpenShift Container Platform 3.4 is expected to be available through the Red Hat Customer Portal. ®

http://www.theregister.co.uk/2017/01/18/red_hat_adorns_openshift_container_platform_with_simplified_storage/

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

Simple Quorum Strive for the FreeBSD CTL HA and the BeaST Storage System

by Mikhail Zakharov

During our experiments on developing the BeaST storage system we faced the lack of an automatic LUN failover function of the CTL HA subsystem. Yes, we can switch LUNs with CTL HA, but we have to do it manually setting “Primary role” to the alive controller: `sysctl`

`kern.camctl.ha_role=0`

We also have to do it fast enough, otherwise, a client host may lose access to the drives of the storage system.

This issue can be solved if we involve an existing HA clustering software like Pacemaker/Corosync or Heartbeat. Although both of these remarkable projects have Linux roots, you can find their skeletons in the FreeBSD port collection (report if you succeed in installing and using them).

Currently, for the BeaST project, we do not need Linux compatibility or any fancy clustering abilities, so I have spent a couple of weeks writing a simple quorum device solution for our experiments with the BeaST, FreeBSD and CTL HA. The result of my work is the Beast Quorum (BQ) version 1.0, which you can download from Sourceforge site.

In order to test the Beast Quorum, I am going to use the same environment I have installed earlier for the BeaST storage and CTL HA experiments. We will run two storage controllers (ctrl-a, ctrl-b) and a client host (cln-1). But now we have to prepare two virtual SAS drives (da0 and da1), which are configured as “shareable” in Virtual Media Manager and simultaneously connected with both storage controllers. The da0 drive is for storing data, and da1 is for shared quorum.

Actually, the BeaST quorum device can be a shared drive, a slice or a partition of almost any size, recognizable by FreeBSD.

These IP addresses are assigned to the hosts:

Host	Private network	Public network
ctrl-a	192.168.10.101	192.168.20.101
ctrl-b	192.168.10.102	192.168.20.102
cln-1	-	192.168.20.103

We are not going to share the quorum drive with clients so there is no need to include corresponding definitions into iSCSI configuration and `/etc/ctld.conf`:

ctrl-a	ctrl-b
<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.10.101 } target iqn.2016-01.local.beast:target0 { auth-group no-authentication portal-group pg0 lun 1 { path /dev/da0 } }</pre>	<pre>portal-group pg0 { discovery-auth-group no-authentication listen 192.168.10.102 } target iqn.2016-01.local.beast:target0 { auth-group no-authentication portal-group pg0 lun 1 { path /dev/da0 } }</pre>

Installation and configuration

Today, BQ is in the early development stage, it supports only basic functions, which are essential to perform failover operations on storage controllers (nodes). Also there is no FreeBSD port for it, or even decent Makefile with “install” section yet. Obviously, we have to perform most of the installation tasks by hands, luckily there are not many things to do.

So, let's boot the storage controllers (ctrl-a, ctrl-b) and run these commands on both of them to download BQ source code and compile it:

```
beast@ctrl-a:~/ % fetch
http://netcologne.dl.sourceforge.net/project/bquorum/bq-1.0.tgz

bq-1.0.tgz                                100% of 6184  B   37
MBps  00m00s

beast@ctrl-a:~/ % tar zxvf bq-1.0.tgz

x bq-1.0/
x bq-1.0/bq.c
x bq-1.0/bq.h
x bq-1.0/bq.trigger.n0
x bq-1.0/bq.trigger.n1
x bq-1.0/README
x bq-1.0/Makefile

beast@ctrl-a:~/ % cd bq-1.0

beast@ctrl-a:~/bq-1.0 % make

cc -O2 -pipe -fstack-protector -fno-strict-aliasing -Wall -o bq bq.c
```

As everything has come off well, we can install the quorum device. Therefore, gain root access and run:

```
root@ctrl-a:/home/beast/bq-1.0 # ./bq -I -d /dev/dal -f 1 -t 10

BeaST Quorum is installed.
```


Where:

- l means “install” command;
- d specifies shared device;
- f is heartbeat frequency, default is 1 second;
- t heartbeat timeout in seconds, default is 10 seconds.

Now we have configured BQ nodes to send heartbeats every second. Each node monitors these heartbeats and if the other node is silent for the period of 10 seconds, it means that the node is dead.

After quorum installation, we can check BQ shared drive configuration, and -L key is used for this operation:

```
root@ctrl-a:/home/beast/bq-1.0 # ./bq -L -d /dev/dal
```

```
BQuorum Label:          BQ
```

```
BQuorum Version:        32
```

```
Heartbeat Frequency:     1
```

```
Heartbeat Timeout:       10
```

```
Node 0 Current State:    0
```

```
Node 1 Current State:    0
```

```
Node 0 Timestamp block:  1
```

```
Node 1 Timestamp block:  2
```

```
Node 0 Timestamp value:  0
```

```
Node 1 Timestamp value:  0
```

As we can see, the BQ header is installed but both nodes are currently offline: Node States – 0 and Timestamp values (heartbeats counters) – 0.

The command above reads the BeaST header structure from the shared drive, we get the same picture from the ctrl-b:

```
root@ctrl-b:/home/beast/bq-1.0 # ./bq -L -d /dev/da1

BQuorum Label:          BQ
BQuorum Version:        32
Heartbeat Frequency:     1
Heartbeat Timeout:       10
Node 0 Current State:    0
Node 1 Current State:    0
Node 0 Timestamp block:  1
Node 1 Timestamp block:  2
Node 0 Timestamp value:  0
Node 1 Timestamp value:  0
```

The same information from both controllers is a sign that the quorum header is installed correctly and both nodes are ready to run BQ operations. Now let's start BQ on both controllers in daemon mode:

```
root@ctrl-a:/home/beast/bq-1.0 # ./bq -S -d /dev/da1 -n 0 -s
/home/beast/bq-1.0/bq.trigger.n0 -l /var/log/bq.log

root@ctrl-b:/home/beast/bq-1.0 # ./bq -S -d /dev/da1 -n 1 -s
/home/beast/bq-1.0/bq.trigger.n1 -l /var/log/bq.log
```

Where:

-S means "start" quorum operations;

-n Current node id 0 or 1;

- s command (trigger) to run on node alive/dead event;
- l path to the BQ log file.

The most interesting key here is -s, which specifies a trigger – a command or a script to run when BQ detects a change in the state of any node:

Currently, BQ on each node recognizes three states:

Node	State	State number
Current (this) node	alive	1
Other (that) node	dead	2
Other (that) node	alive	3

On these states, BQ runs trigger command with three parameters: “reporting node id”, “node, which caused the trigger to run”, and “the new state”.

For example, in our case, BQ can start a trigger like this:

```
home/beast/bq-1.0/bq.trigger.n0 0 1 dead
```

And it means: Node 0 reports that Node 1 is now Dead.

For our tests, I have written simple Bourne shell scripts (they are also included into BQ sources) to handle two main states, 1 and 2.

So we have two similar scripts bq.trigger.n0 and bq.trigger.n1 for each node. And the only difference between them is in the IP addresses of the nodes:

```
root@ctrl-a:/home/beast/bq-1.0 # diff bq.trigger.n0 bq.trigger.n1
39,40c39,40
< this_node="192.168.10.101:7940"
< that_node="192.168.10.102:7940"
```

```
---  
  
> this_node="192.168.10.102:7940"  
  
> that_node="192.168.10.101:7940"
```

These trigger scripts handle node failover operations manipulating ctld and setting proper `kern.camctl.ha_role` values on the controllers.

After both nodes have been started, let's once again check BQ status with -L key:

```
root@ctrl-a:/home/beast/bq-1.0 # ./bq -L -d /dev/dal  
  
BQuorum Label:          BQ  
  
BQuorum Version:        32  
  
Heartbeat Frequency:    1  
  
Heartbeat Timeout:      10  
  
Node 0 Current State:   1  
  
Node 1 Current State:   1  
  
Node 0 Timestamp block: 1  
  
Node 1 Timestamp block: 2  
  
Node 0 Timestamp value: 1480787008  
  
Node 1 Timestamp value: 1480787008
```

Now both nodes are alive (Node Current States) – 1, and the timestamps are updated every second.

So everything works for now and we can proceed with the client configuration. Lets start our cln-1 client and connect it with remote iSCSI drives:

```
root@cln-1:/ # sysctl kern.iscsi.fail_on_disconnection=1  
  
root@cln-1:/ # iscsictl -A -p 192.168.20.101 -t  
iqn.2016-01.local.beast:target0
```



```
root@cln-1:/ # iscsictl -A -p 192.168.20.102 -t
iqn.2016-01.local.beast:target0

root@cln-1:/ # gmultipath label -A HA /dev/da0 /dev/da1

root@cln-1:/ # newfs /dev/multipath/HA

root@cln-1:/ # mount /dev/multipath/HA /mnt
```

Most of the commands on the client were discussed in the previous articles, but there is one exception: `gmultipath` with “label” command writes multipathing metadata to the drives. It allows multipath to restore paths when drives are getting back.

Failover functional tests

Like in previous tests, we can push the whole construction to work by continuously copying a file:

```
# while true; do cp ports.tar.gz /mnt; done
```

Then start gathering statistics from the client and nodes:

```
root@cln-1:/ # iostat -xd 5 | grep '^d'

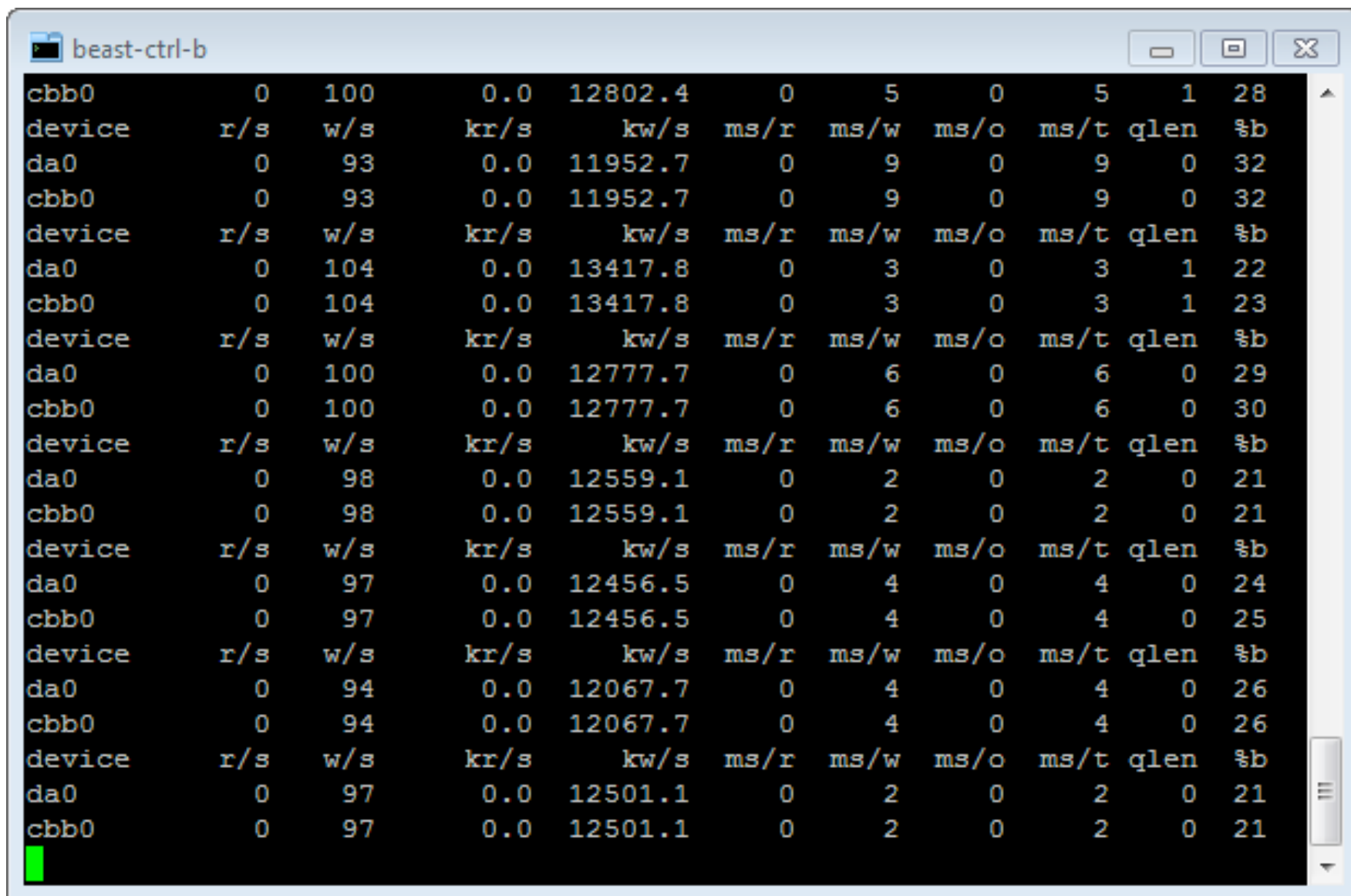
root@ctrl-a:/ # iostat -xd 5 | egrep '^dev|^da0|^cb'

root@ctrl-b:/ # iostat -xd 5 | egrep '^dev|^da0|^cb'
```

So all paths and a physical drive as well are utilized. See the appropriate screenshots below.

beast-cln-1										
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	93	0.0	11804.2	0	37	0	37	0	98
da1	0	94	0.0	12045.1	0	36	0	36	1	97
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	101	0.0	12894.7	0	31	0	31	1	97
da1	0	97	0.0	12481.6	0	32	0	32	0	97
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	101	0.0	12909.1	0	33	0	33	1	97
da1	0	106	0.0	13538.5	0	32	0	32	0	98
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	102	0.0	13074.9	0	35	0	35	0	98
da1	0	97	0.0	12409.7	0	38	0	38	1	97
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	105	0.0	13383.1	0	29	0	29	1	98
da1	0	102	0.0	13139.0	0	31	0	31	2	96
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	101	0.0	12978.0	0	34	0	34	1	98
da1	0	105	0.0	13540.1	0	33	0	33	0	98
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	99	0.0	12675.9	0	37	0	37	1	97
da1	0	99	0.0	12714.2	0	37	0	37	0	97

beast-ctrl-a										
da0	0	105	0.0	13391.2	0	6	0	6	0	30
cbb0	0	105	0.0	13391.2	0	6	0	6	0	31
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	105	0.0	13467.4	0	2	0	2	1	24
cbb0	0	105	0.0	13467.4	0	2	0	2	1	24
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	102	0.0	13035.9	0	3	0	3	0	27
cbb0	0	102	0.0	13035.9	0	3	0	3	0	27
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	102	0.0	13021.5	0	2	0	2	0	26
cbb0	0	102	0.0	13021.5	0	2	0	2	0	26
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	102	0.0	12966.7	0	3	0	3	0	28
cbb0	0	102	0.0	12966.7	0	3	0	3	0	28
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	105	0.0	13417.9	0	2	0	2	0	26
cbb0	0	105	0.0	13417.9	0	2	0	2	0	26
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	106	0.0	13604.3	0	2	0	2	1	24
cbb0	0	106	0.0	13604.3	0	2	0	2	1	25



cbb0	0	100	0.0	12802.4	0	5	0	5	1	28
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	93	0.0	11952.7	0	9	0	9	0	32
cbb0	0	93	0.0	11952.7	0	9	0	9	0	32
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	104	0.0	13417.8	0	3	0	3	1	22
cbb0	0	104	0.0	13417.8	0	3	0	3	1	23
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	100	0.0	12777.7	0	6	0	6	0	29
cbb0	0	100	0.0	12777.7	0	6	0	6	0	30
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	98	0.0	12559.1	0	2	0	2	0	21
cbb0	0	98	0.0	12559.1	0	2	0	2	0	21
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	97	0.0	12456.5	0	4	0	4	0	24
cbb0	0	97	0.0	12456.5	0	4	0	4	0	25
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	94	0.0	12067.7	0	4	0	4	0	26
cbb0	0	94	0.0	12067.7	0	4	0	4	0	26
device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	97	0.0	12501.1	0	2	0	2	0	21
cbb0	0	97	0.0	12501.1	0	2	0	2	0	21

As the traffic is going through both controllers, let's find the "Primary" CTL HA node:

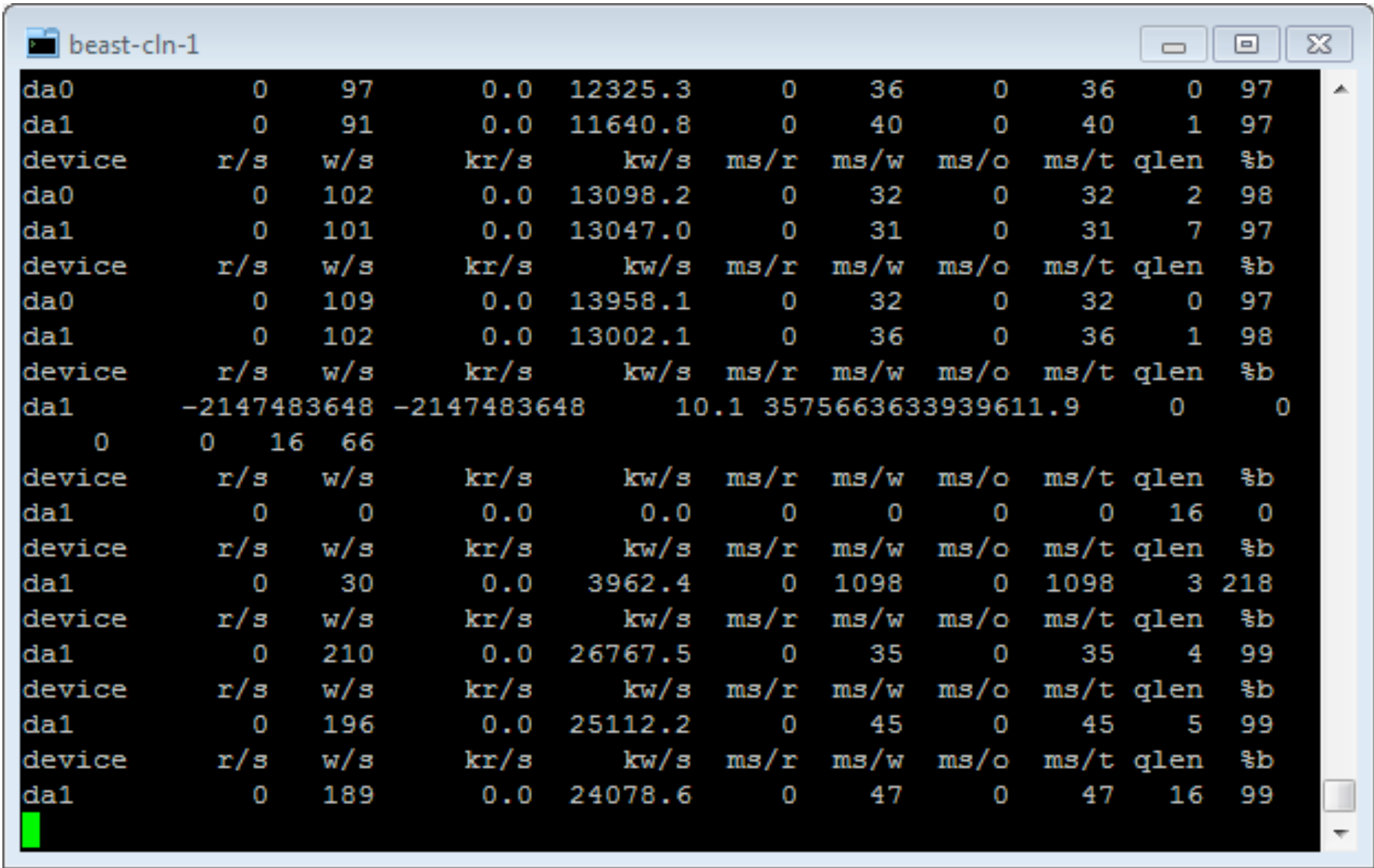
```
root@ctrl-a:/ # sysctl kern.camctl.ha_role
kern.camctl.ha_role: 0
root@ctrl-b: # sysctl kern.camctl.ha_role
kern.camctl.ha_role: 1
```

We can see the "Primary" CTL HA node is ctrl-a. We remember that bare CTL HA needs user interaction to change HA role and make controller failover. Let's reboot ctrl-a controller and see if the Beast Quorum can automatically do the job and failover to ctrl-b:

```
root@ctrl-a:/ # reboot
```

Below there are screenshots from cln-1 and ctrl-b which I captured just a few second after ctrl-a's disappearance.

As we can see, the cln-1 client lost access to the ctrl-a drive (da0), and the traffic stopped on (da1) but when BQ turned on CTL HA “Primary” role on the ctrl-b, the traffic began to run again:



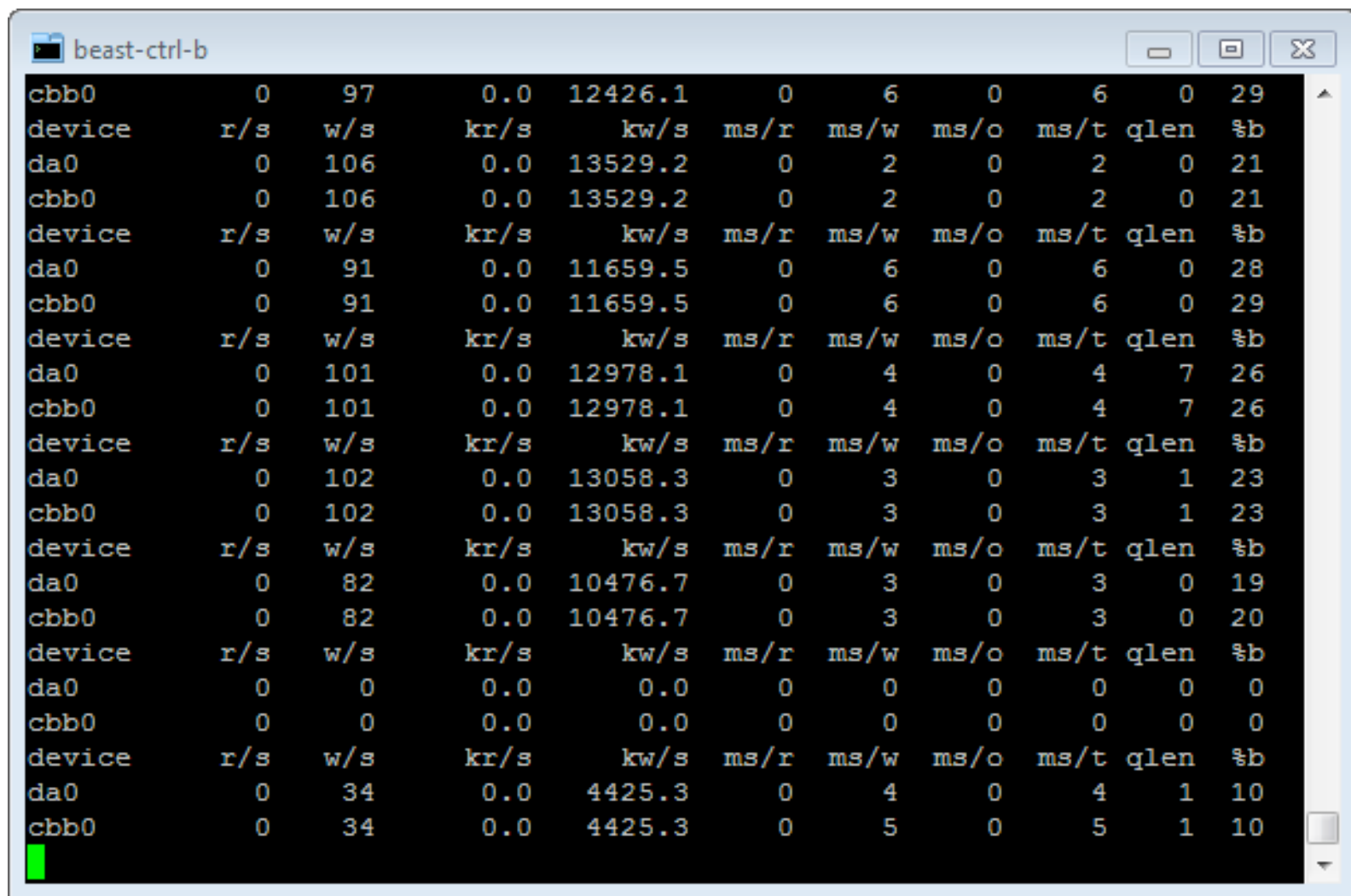
The corresponding picture from ctrl-b:

We can state, BQ correctly detects controllers death and automatically switches node roles.

Now let’s test what happens when the node comes back.

Therefore, boot the ctrl-a again and start BQ on it:

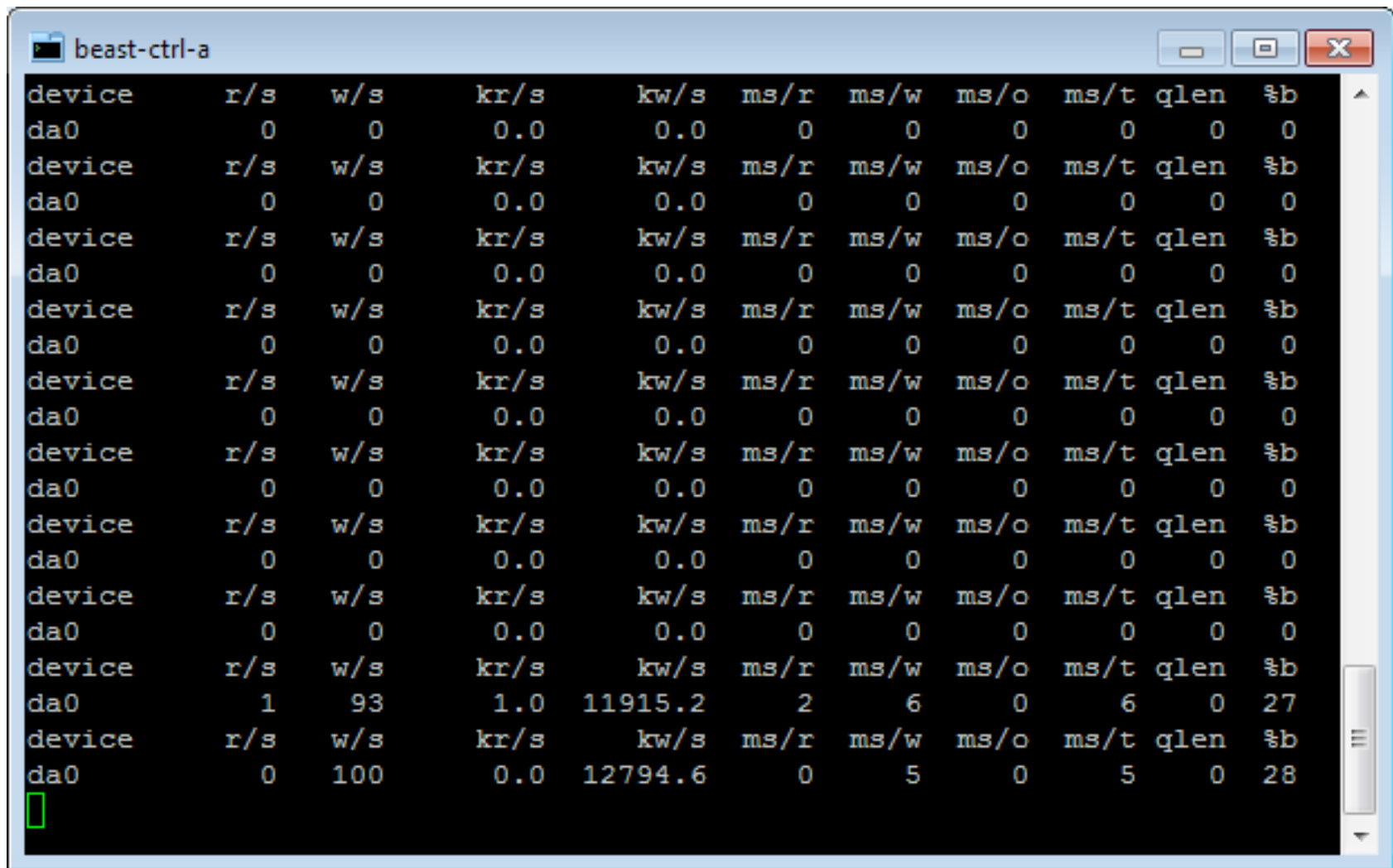
```
root@ctrl-a:/home/beast/bq-1.0 # ./bq -S -d /dev/da1 -n 0 -s
/home/beast/bq-1.0/bq.trigger.n0 -l /var/log/bq.log
```

The screenshot shows a terminal window titled "beast-ctrl-b" displaying a series of disk performance metrics. The metrics are organized into groups for two devices: da0 and cbb0. Each group includes a header row with labels (device, r/s, w/s, kr/s, kw/s, ms/r, ms/w, ms/o, ms/t, qlen, %b) followed by two rows of data. The data shows that da0 is receiving traffic, while cbb0 is not.

Device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
cbb0	0	97	0.0	12426.1	0	6	0	6	0	29
da0	0	106	0.0	13529.2	0	2	0	2	0	21
cbb0	0	106	0.0	13529.2	0	2	0	2	0	21
da0	0	91	0.0	11659.5	0	6	0	6	0	28
cbb0	0	91	0.0	11659.5	0	6	0	6	0	29
da0	0	101	0.0	12978.1	0	4	0	4	7	26
cbb0	0	101	0.0	12978.1	0	4	0	4	7	26
da0	0	102	0.0	13058.3	0	3	0	3	1	23
cbb0	0	102	0.0	13058.3	0	3	0	3	1	23
da0	0	82	0.0	10476.7	0	3	0	3	0	19
cbb0	0	82	0.0	10476.7	0	3	0	3	0	20
da0	0	0	0.0	0.0	0	0	0	0	0	0
cbb0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	34	0.0	4425.3	0	4	0	4	1	10
cbb0	0	34	0.0	4425.3	0	5	0	5	1	10

And the BeaST Quorum has done everything right to put the node online. We can see that da0 drive on the ctrl-a begins to receive traffic:



The screenshot shows a terminal window titled "beast-ctrl-a" displaying disk performance metrics for the da0 device. The metrics are organized into groups, each with a header row and two rows of data. The data shows that da0 is receiving traffic, with the last two rows showing non-zero values for r/s, w/s, kr/s, kw/s, ms/r, ms/w, ms/o, ms/t, qlen, and %b.

Device	r/s	w/s	kr/s	kw/s	ms/r	ms/w	ms/o	ms/t	qlen	%b
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	0	0	0.0	0.0	0	0	0	0	0	0
da0	1	93	1.0	11915.2	2	6	0	6	0	27
da0	0	100	0.0	12794.6	0	5	0	5	0	28

Multipathing on the cln-1 client also works well and we can see da0 is online again:

```
beast-cln-1
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     109    0.0    13975.4    0      29      0      29      1     96
da0       0     104    0.0    13271.6    0      32      0      32      0     97
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     104    0.0    13288.7    0      33      0      33      3     98
da0       0     105    0.0    13492.5    0      32      0      32      1     97
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     107    0.0    13695.6    0      32      0      32      1     97
da0       0     103    0.0    13221.5    0      35      0      35      2     98
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     104    0.0    13284.6    0      32      0      32      0     97
da0       0     107    0.0    13725.7    0      32      0      32      1     97
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     107    0.0    13731.2    0      33      0      33      1     97
da0       0     101    0.0    12948.8    0      36      0      36      0     97
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0      88    0.0    11245.4    0      40      0      40      2     97
da0       0      90    0.0    11603.2    0      40      0      40      0     98
device    r/s    w/s    kr/s    kw/s    ms/r    ms/w    ms/o    ms/t    qlen    %b
da1       0     107    0.0    13657.3    0      31      0      31      1     98
da0       0     100    0.0    12826.1    0      33      0      33      0     97
```

Conclusions

Yes, it works! The BeaST Quorum successfully does the job for automatic CTL HA controller/node failover operations. Without BQ we have to do it manually, that is nonsense for the reliable storage systems.

But there are a lot of things to do to improve BQ. For example, we must take care of reliable mechanisms of restarting BQ daemon if it has been occasionally killed while the controller continues to run; add failover support for individual LUNs; improve heartbeat processing and implement sliding redundant heartbeat blocks; fix log procedures to prevent simultaneous writes into the same file; and much more, including typographic errors.

Note the BeaST Quorum is currently in the early development stage! Use it for testing purposes



About the Author:

My name is Mikhail E. Zakharov and I am a proud SAN/storage IBMer. 10 years of experience in large SAN and storage environments: mainly Hitachi, HP and Brocade. Empty – expect-like tool author. FreeBSD enthusiast.



FreeBSD

BSD: It Really Never Dies

by Pedro Giffuni

This year, I was one of three developers representing FreeBSD at the Google Summer of Code 2016 Mentor Summit in Sunnyvale, California. It was a nice opportunity to get in touch with other projects and let FreeBSD get a little better known. We don't really have many chances to meet other developers from our own project so during the dinner, we started discussing a variety of matters. While we knew there was someone from NetBSD, we hadn't seen anyone from the other BSD projects. One of my colleagues made a comment along the lines of "perhaps NetBSD should be reconsidering its mere existence," to which I replied ... "that will never happen, pretty much as we will not ever consider closing our project just because another project is more popular."

There is great comradery between the BSD derivatives but for a while FreeBSD has been standing as the variant with the best performance and functionality. Not only does it have more features and better performance (yes, I am admittedly biased), it also has the biggest community. If FreeBSD were to somehow lose all its advantages to another BSD variant or Linux, I can dare say we would still not give up and continue working on FreeBSD. This is not because we are stubborn and if you bear with me, I will try to explain why.

Many outsiders would just think it would be better to have all projects join forces and there would be a valid rationale around Darwin's evolutionism and how the strongest, healthier, community takes over the weakest.

Around 1997, when I started using FreeBSD and open source was in its infancy, “FreeBSD is dying” was a popular joke in Slashdot. Another project I developed for, Apache OpenOffice, has been under attack from its inception, and while I admit it wouldn’t surprise me if it ends up in the attic someday, it still gets around 50 million downloads per release. Open source projects do end their life cycles, but betting on the death of a project is something that will likely end up in embarrassment.

A sane environment

Theoretically, concentrating all the resources in one huge development umbrella is more efficient. That is the idea behind all the corporate mergers, but it usually doesn’t work very well if we are to trust cases like Hewlett Packard. HP was a great innovator in all technological areas and I was huge fan of their calculators, until the moment where they bought Compaq, which itself had already swallowed Digital Equipment Corporation. Nowadays, HP doesn’t produce calculators anymore and has been struggling to sell their PC business.

Well, either Darwin was wrong or we simply haven’t understood what he was talking about. Darwinism, in every sense, has been revised by history. If it is survival of the fittest, the fittest is not the strongest on its own (ask the dinosaurs); in software development understanding team work and niche markets are essential.

In the BSDs, we have the three big variants we all like and the code is openly exchanged between the three of them. FreeBSD occasionally merges features from both NetBSD and OpenBSD; true security fixes always find their way in all BSD variants very fast and we may end up working together on many features. The ext2fs implementation, which I have maintained for some years in FreeBSD, is very similar to the base UFS used in all the BSDs and has huge parts of it that came from NetBSD; ultimately the ext4 read-only support from FreeBSD has found its way into both NetBSD and OpenBSD. Wouldn’t it be easier to just do the development in the same tree?

Ext2fs is not really an important filesystem in the BSDs. In the main filesystem area, which is historically UFS, FreeBSD developers chose to bet on an innovative solution, soft updates, versus the more common journaling. NetBSD chose journaling. Having a merged team might have meant either one of the developers forgetting about their approach in favor of the one that had more warm bodies involved. Curiously, Linux developers were also interested in soft-updates but perceived it was too complex for them to cope with. In this case, as in other cases involving all projects, having a small team focused on specific things brings better results, such as having both UFS with soft-updates and ZFS, that having a larger group introducing their own ideas. If we also had the influence of OpenBSD developers, that would have meant a “flamewar” around ZFS.

Being the biggest BSD, FreeBSD certainly has limited resources. We always want new blood and

C developers are starting to get less common so you would think that getting all the developers we can should be critical. Instead, the social skills and the ability to work in a team are seen as essential to the project. The BSD projects scale as well as the communities supporting them; becoming a FreeBSD developer has barriers for entry and we like it that way.

If developers lack their social abilities or their technical solution doesn't fit well with the existing development, developers can always fork, therefore avoiding conflict. This, IMHO, is not wrong at all. Forks are good. I am personally very fond of TrueOS, FreeNAS and pfSense; they are truly innovative forks of FreeBSD and they are doing very nice things that fall out of the scope of FreeBSD as a general-purpose OS, which doesn't mean we can't learn from them. I am personally not very fond of HardenedBSD, but they are certainly fine people that believe in what they do and we can eventually learn something, one way or another, from them.

We are also seeing some commercial forks of FreeBSD taking over the headlines. It is arguable how much such efforts contribute back to open source development. I was recently asked if Apple has been contributing back to FreeBSD, to which I answered: "Oh yes, they don't take the time to send us back their changes but they do make most of their interesting developments available, so we have candidly taken things like chunks of libc and their wonderful compiler". In the case of SONY, we have seen much less but the fact that they are finding the code useful for their hugely successful PS4 is personally a great satisfaction.

Development: it goes on

Understanding the differences and the needs for them, is not really sufficient to justify that a project may die or not; it's ultimately the community who decides how far a project can advance.

Here is a personal listing on why I think BSD projects will continue advancing:

- Companies: money is important and necessary. The permissive and commercially friendly license behind the BSDs is a warranty that development will go on. We all know about big or mid-sized companies that contribute to FreeBSD's development, from Yahoo to EMC, Netflix or even Microsoft, companies that give back to the community hiring developers and contributing code, or companies that could do much better like Apple or SONY, they all contribute, keeping the code alive.
- Conferences: even in the modern era of the Internet, where most development happens online, having the chance to meet other developers and discuss new ideas is critical for any consistent software development. Three continents see major BSD conferences every year and the events don't discriminate against any variant. I will openly state this, please donate to the FreeBSD Foundation, their sponsorship for such events help support significantly all the BSDs in significant ways.

- New software development tools: The BSDs tend to be slow to change not only in the code base, where stability tends to be more important than experimenting heavily, but in the development methods. C is still a fine language for an Operating System and the use of Version Control from the early days has been key to have a consistent development model. It so happens that at least FreeBSD has had to evolve significantly incorporating new tools for the development process. Unit testing, continuous Integration, static code checking and code review software are all part of FreeBSD and are extending rapidly to the other BSDs.
- Competitiveness: no project solves everything for everyone. As long as the other open source alternatives out there don't cover the needs of every potential user, there will be space for alternative Operating Systems: this is normal. On the other hand, even if there were an Operating System that does everything as well or slightly better than what your favorite BSD variant does, it's not a given that such system should be used, this is not only normal, in the case of security having such monocultures can be fatal. Some companies have found that having a mixture of BSD and Linux-based OSs improve the reliability of their service, especially in the widespread of attacks targeting the most popular Linux -based systems. As of lately systemd has become a good reason to embrace the BSDs.

Something that ties together many of the above points is the existence of a foundation to coordinate resources and work among developers and the interested communities. The three main BSD variants have their own non-profit foundations to support them and they are very useful.

I didn't mention diversity. It is sad in a certain sense as the population is composed of many different individuals, but most open source projects are composed mainly of white males (Disclaimer: I am one). The issue is one that worries many communities and it is not something unique to the BSDs. I will make the bold statement that diversity as such is not important, what is important is to be open to diversity. We can't really force the population to be diverse but it is important to stop bullying, harassment and other attitudes that are not acceptable in any community. FreeBSD's code of conduct is certainly a step in the right direction.

Still, all projects die, don't they?

I have sort of explained the reasons why I see development of the major BSDs continue undeterred. Everything comes to an end, and the BSDs are not exempt to extraordinary events, like the "getting hit by a bus"; people come and go and developers are all volunteers. We do what we can, and we constantly recruit valuable people, but there is no guarantee we will get to replace someone or get wonderful maintainers to cover all the code we maintain. We do what we do with love, or at least because we like it, independently of how much we are paid to do it. I will sum up by stating that the main reason why development in the BSDs happens is because each of us is convinced we are doing things right.

Like in any project, the result will be what our developers and contributors want to make out of it. Unlike other projects, which depend on independent developers for specific parts of the system (kernel, libc, userland), in the BSDs it is one team taking care of everything. What you get in the BSDs is necessarily a consistent OS developed by a unique team. The situation is similar to a living body, where individual cells get replaced, constantly evolving but keeping their identity, so that the body as a whole outlives all the cells. FreeBSD, at least, is very likely to outlive me.

About the Author:

Pedro is an Italian born in Bogotá, Colombia (yes, scary) and started working with computers as a teenager with a TRS-80 Color Computer. Having a computer at home in the 1980s was already unusual and weird but using OS-9 Pascal was perhaps too much. Such initial experience made Pedro opt for something different than computers so he graduated from Mechanical Engineering and has been working in Manufacturing and Engineering Quality since then. Therapy and a lot of rest brought him back to the dark world of computers and then in 1987 he met FreeBSD. He has been a speaker in EuroBSDCon 2008, BSDCan 2014 and ApacheCon EU 2012, 2014, and 2016.

The State of Multimedia in OpenBSD

by David Carlier

OpenBSD is often seen as an excellent operating system for routers, in which it indeed excels.

In this article, we will see that OpenBSD can have more use cases, thus being used for sound editing, video games or multimedia at large.

1. What is available and how?

Most of the software require some local patching, either to support the particular audio subsystem or basically to adapt the code to OpenBSD. Despite being slower than Linux and FreeBSD, the performances in general are acceptable enough to be enjoyable under OpenBSD. The good news is the number of software available under OpenBSD ports/binary packages has the tendency to grow. Some missing features, like the full path retrieval of an executable or the lack of real time signal functions, for example, can block some portage, though, as we'll see below.

2. Audio

sndio audio system, shipped with OpenBSD since 2008, is basically the intermediate between the audio hardware and the applications. There is also a daemon in the pure OpenBSD's fashion named sndiod.

As such, a large amount of audio related software contains a sndio driver, and whether it is shipped originally with or not, the amount of patches merged upstream increases over the time. Here is a list of the supported packages:

- Audacity, the popular audio waveform editor.
- Audacious, a powerful audio player.
- LMMS, very complete music platform production (despite the actual version is fairly old, a new

version will be released around the beginning of 2017).

- hydrogen, to make drum patterns.
- Timidity, the famous synthesizer.
- OpenAL, a popular audio library for games and even multimedia at large with sndio partly supported (no recording feature for the moment).
- Speex, which serves for various First Person Shooters.

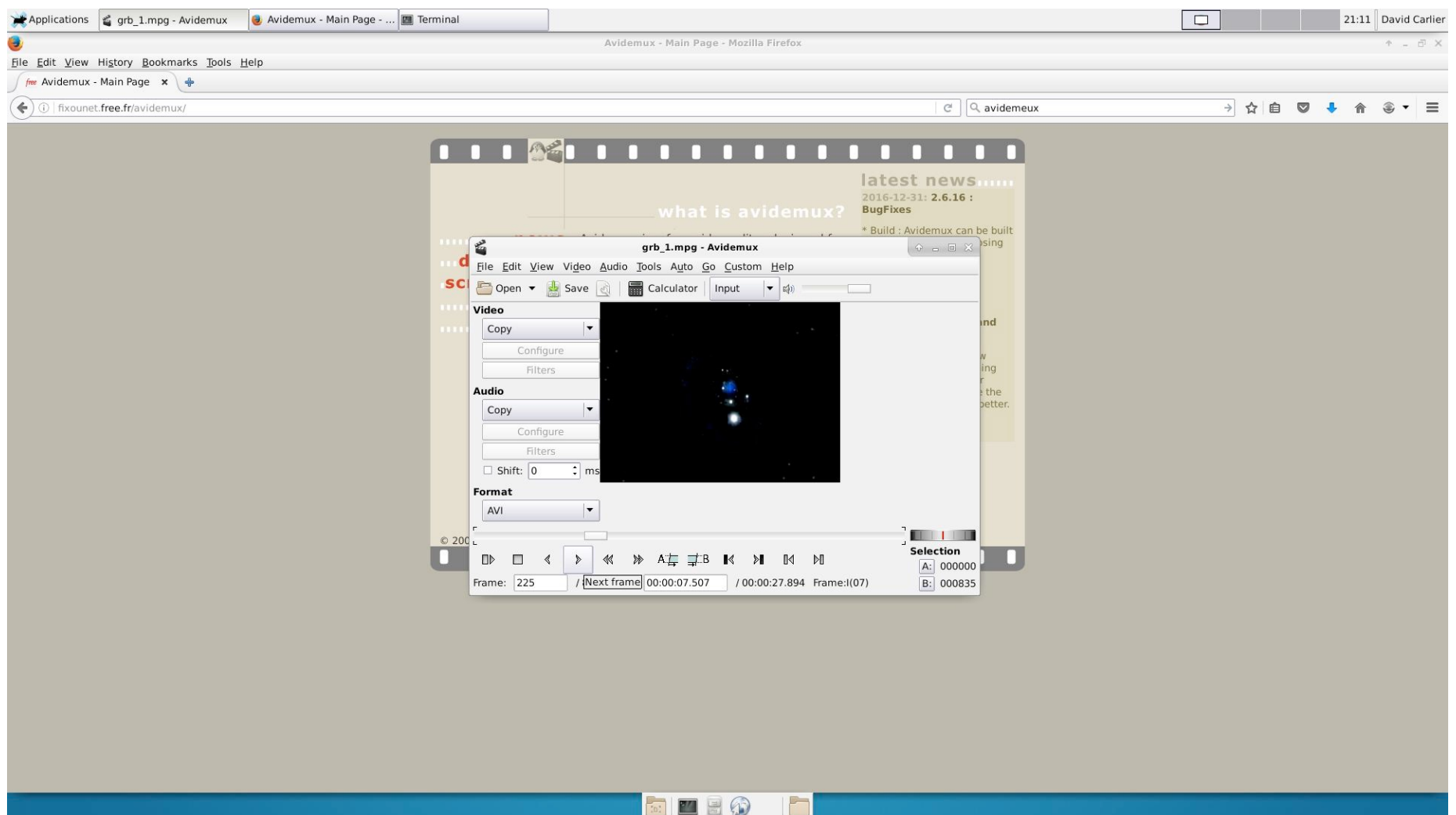


Illustration 1: Audacious works fairly fine on OpenBSD

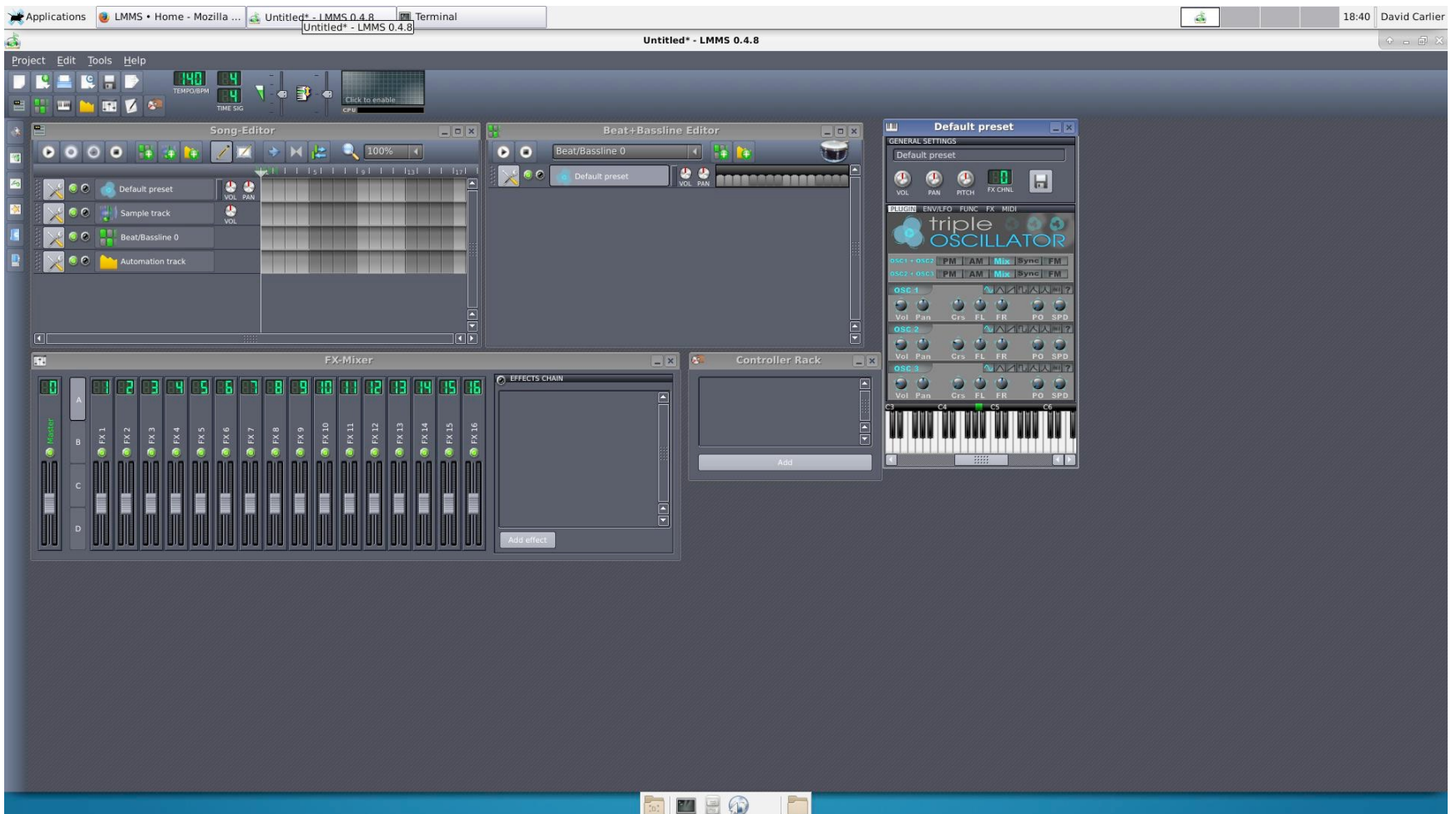


Illustration 2: LMMS on OpenBSD is an old version, but a real version update might be scheduled with full sndio support and the virtual synthesizer above playable with keyboard.

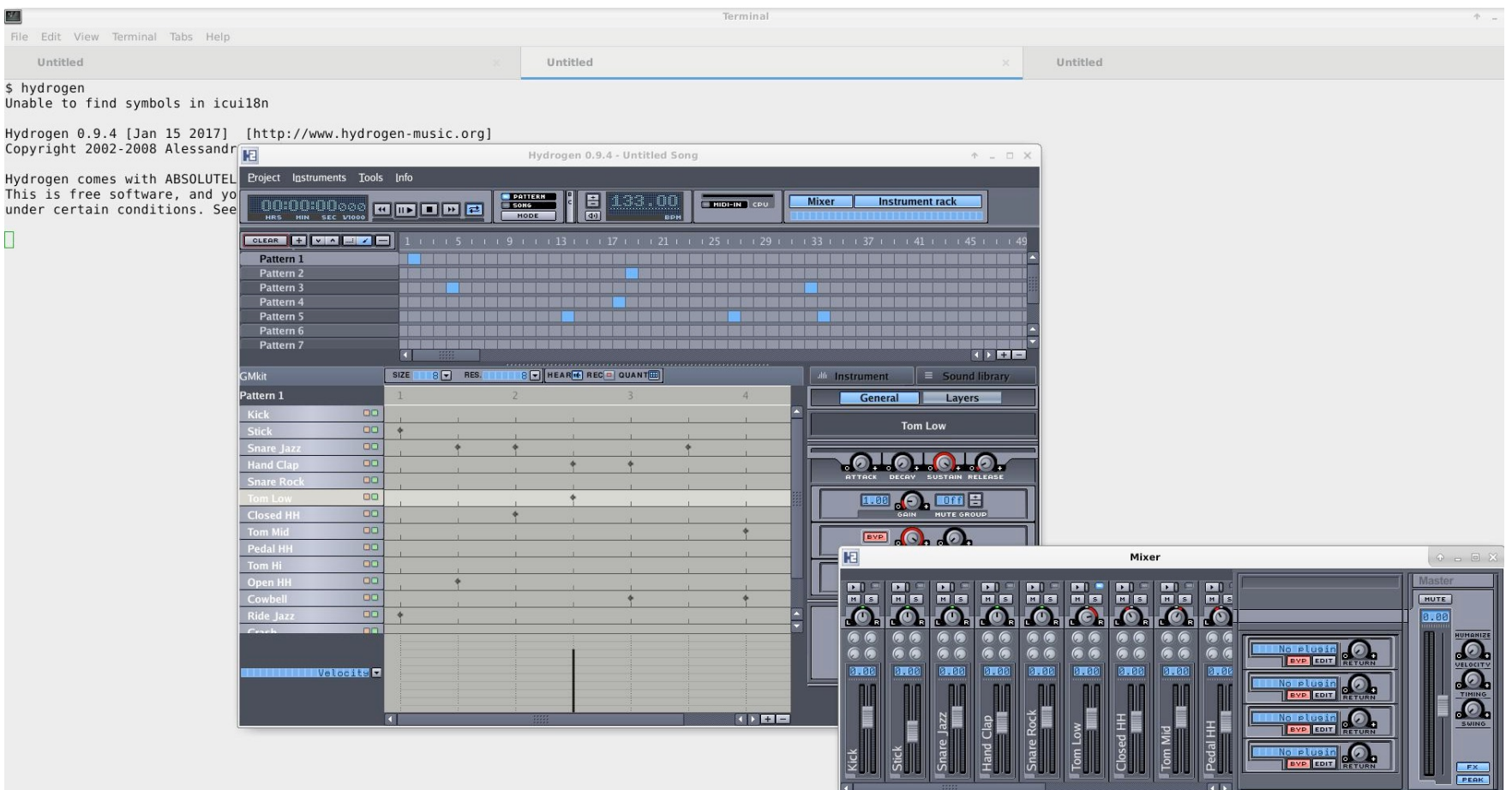


Illustration 3: Hydrogen, perfect for making drum patterns for electronic music.

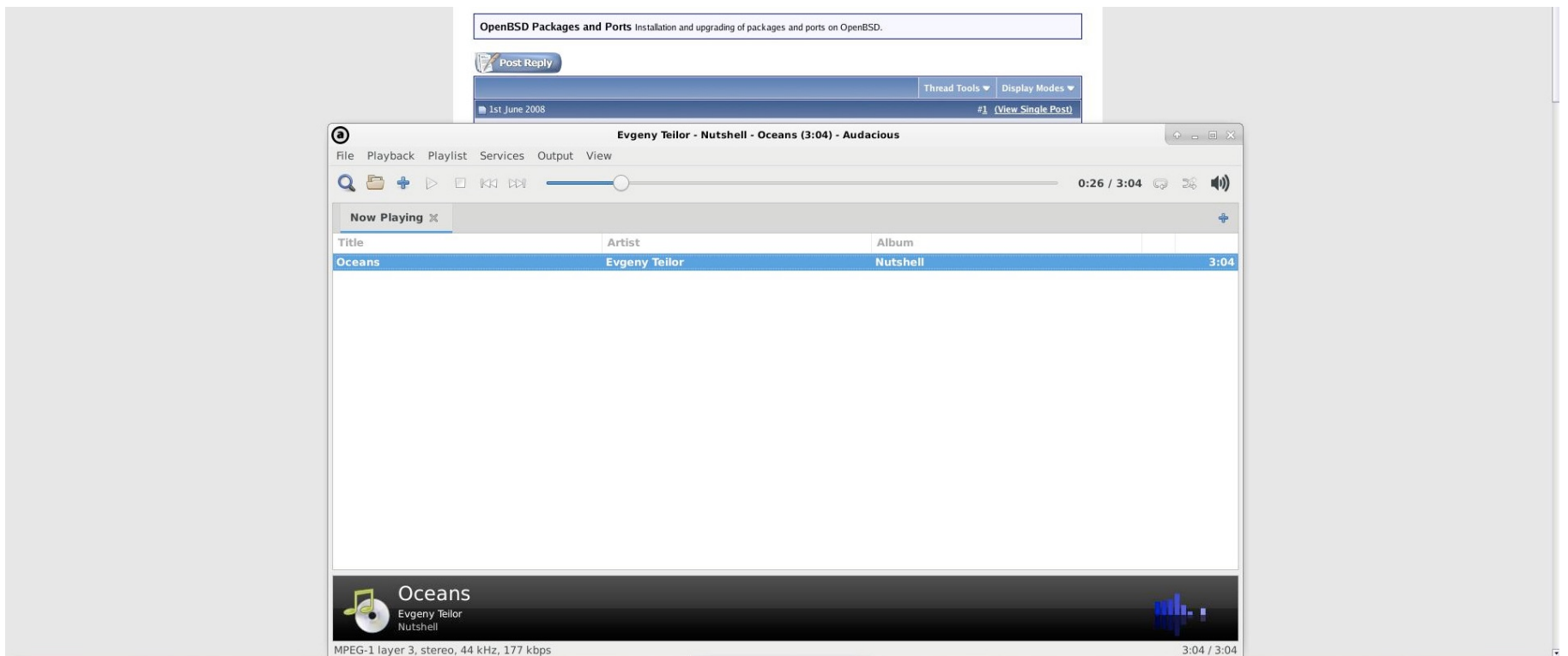


Illustration 4: avidemux, to edit videos.

For sure, the musical platform softwares cannot really compete with their commercial counterparts (except maybe Ardour) but for an amateur need, any user can make decent musical productions.

3. Video games

There is a reasonable amount of video games, First Person Shooters, racing, strategy, board

- yquake2 which is nothing less than one (very good) Quake II implementation;
- dhewm3 a client for Doom III;
- supertuxkart very popular super Mario Kart like;
- lugaru, a 3d combat game with rabbit works directly well on OpenBSD;
- DustRacing2d, an old-school 2d racing game, likewise works without modification on OpenBSD.
- OpenMw, a free implementation of Morrowind client.
- OpenTTD, a free implementation of Transport Tycoon Deluxe.

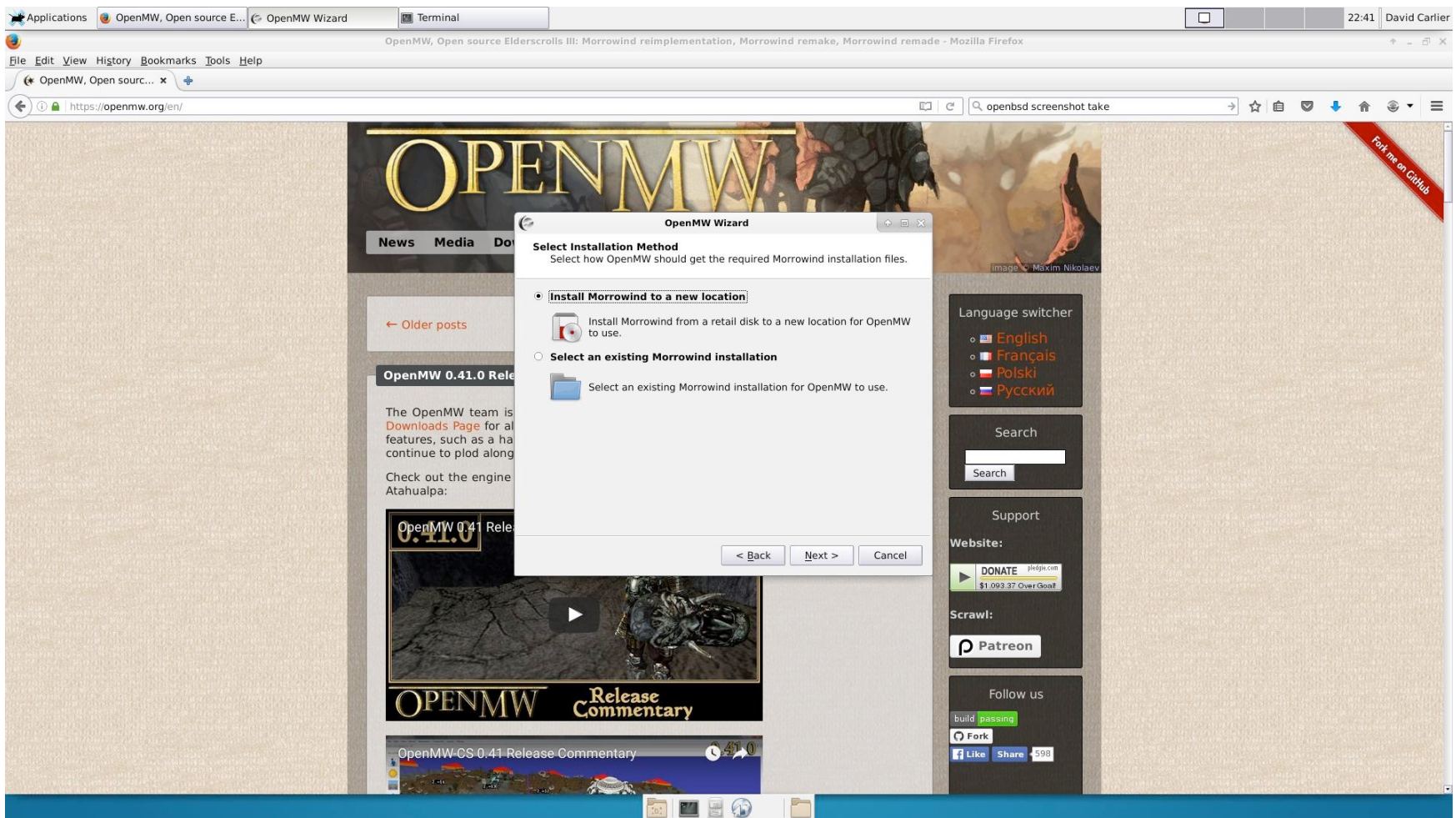


Illustration 5: OpenMW works perfectly on OpenBSD.

Again, the video games cannot defeat most of commercial video games but they bring enough fun for most of users in a decent hardware.

4. For next

Ardour, the famous professional quality audio production platform, could be possibly ported to OpenBSD.

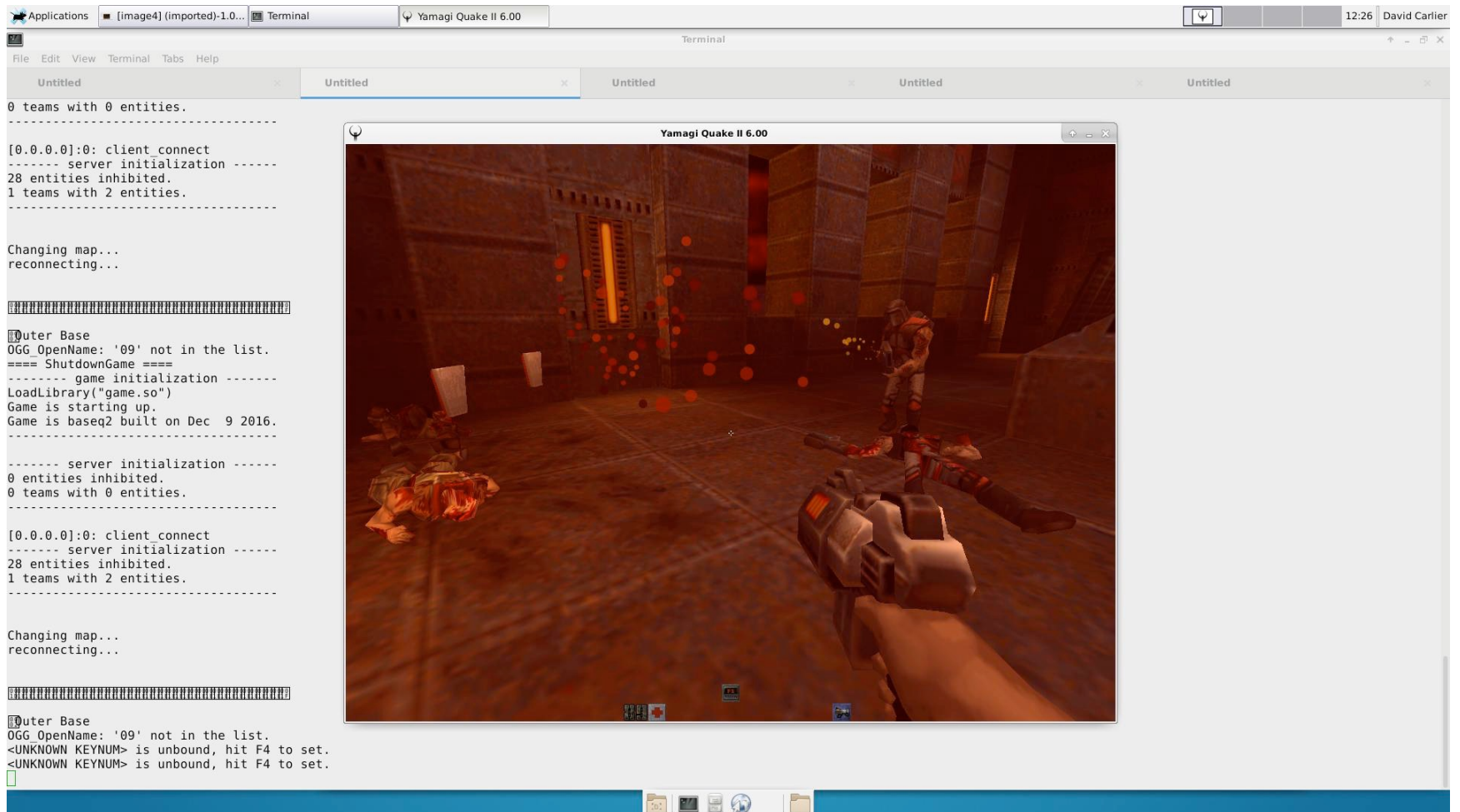
Hopefully, there are couple of new video games which can be possibly ported to OpenBSD :

- OpenSpades, a free implementation of Ace of Spades supports fully OpenBSD.
- SecretChronicles, another SuperMario like (needs CeGI and DeViL libraries but should work well in theory).
- Xonotic, a fast paced FPS working fine with reasonable hardware (should work out of the box).

There are some which would require a certain amount to work, not saying it is impossible but not doable in the near future.

- Unvanquished, the famous pretty FPS based on the not less famous tremendous (basically pNacl which it relies for the maps and so on ... need to be ported to BSD platforms, the game co-debase itself seems portable enough).

- OpenRCT2, which is a free implementation of Roller Coaster Tycoon 2 client (would requires a missing feature in kernel side, the full path retrieval of an executable).
- Falltergeist, based on fallout 2 (same reason as above).



||-

Illustration 7: OpenSpades recently got better support for OpenBSD.

5. Conclusion

There are still work to do, a long way to go ... But for now, OpenBSD offers already a lot of possibilities as you could see in term of multimedia. I wish this article gave you the curiosity to give it a try.

References

OpenSpades <https://github.com/yvt/openspades>

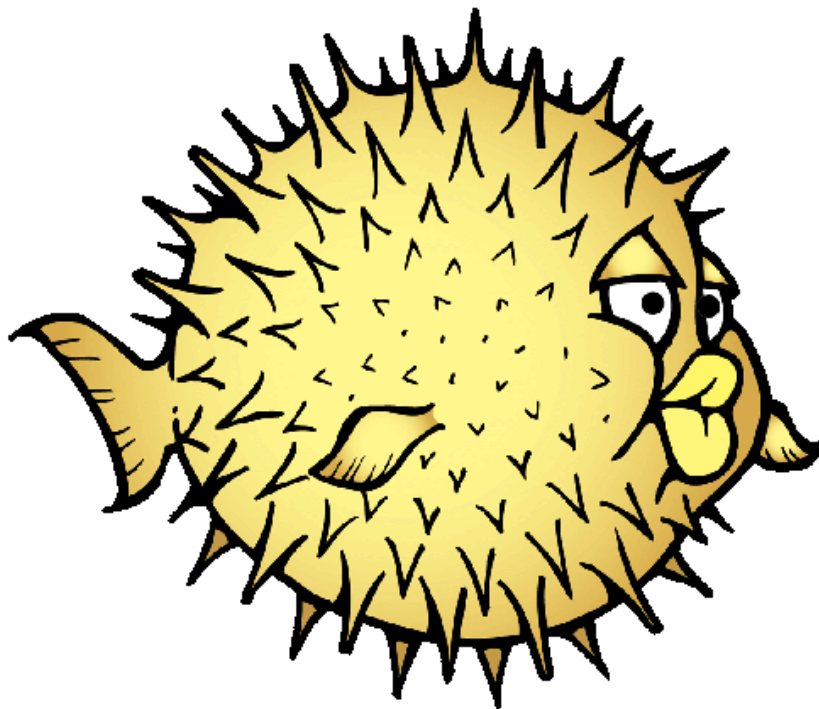
SecretChronicles <https://github.com/Secretchronicles/TSC>

Xonotic <https://github.com/xonotic/xonotic>



About the Author:

David Carlier is a developer since 2001, mainly C/ C++, living and working in Ireland mainly since 2012. He contributes to some open source projects and uses in a daily basis various operating systems mainly BSD flavours.



*Open*BSD

OpenBSD 6.0 Security Feature Changes and Pledge** Security Interface

by Albert Hui

The OpenBSD operating system is well known to be security focused. For the 6.0 release, a noteworthy change is that `systrace`, support for Linux emulation and the `usermount` option has been removed.* If interested, I highly recommend you watch Theo de Raddt's excellent talk at dotSecurity 2016.**

As a replacement for `systrace`, the `pledge(2)` interface allows for fine tuned sandboxing of a process's access to system resources, such as file descriptors/device files, pipes and sockets. This interface mechanism has been implemented at the kernel system call level.

To implement the `pledge` interface in C, you must add reference to the `unistd.h` include statement as follows:

```
#include <unistd.h>
```

and also implement the `pledge` function prototype:

```
int  
  
pledge(const char *promises, const char *paths[]);
```

Several system calls used by OpenBSD already have default restrictions being applied to them. Please refer to `pledge**` for the current list of already restricted system calls.

*<https://www.openbsd.org/60.html>

**<http://www.thedotpost.com/2016/05/theo-de-raadt-privilege-separation-and-pledge>

To implement the pledge interface requires at a minimum a promises argument parameter. The promises argument parameter specifies supported function library reference.

```
if (pledge("stdio", NULL) == -1) {  
    err(1, "pledge");  
}
```

Multiple promise arguments specifiers are also supported by denoting a white space between additional parameters:

```
if (pledge("stdio inet ps", NULL) == -1) {  
    err(1, "pledge");  
}
```

It is of importance to note that subsequent innovations to `pledge()` will only make it more restrictive in terms of functionality and it cannot be reverted to its former less restrictive state. Please consult the OpenBSD manpages `pledge***` for details pertaining to the scope and the list of function call restrictions.

The optional secondary pledge function paths parameter when passed using a NULL value will not change the current value, when a non NULL value has been specified for the paths parameter.

When a process violates the implemented pledge promise by attempting to perform a restricted operation, the process will be automatically killed and core file dump will be created, if possible. The core file dump can be examined to manually use GDB to determine which function calls violated the defined pledge promise.

Additionally, support for the pledge interface has been ported to many other programming languages.****

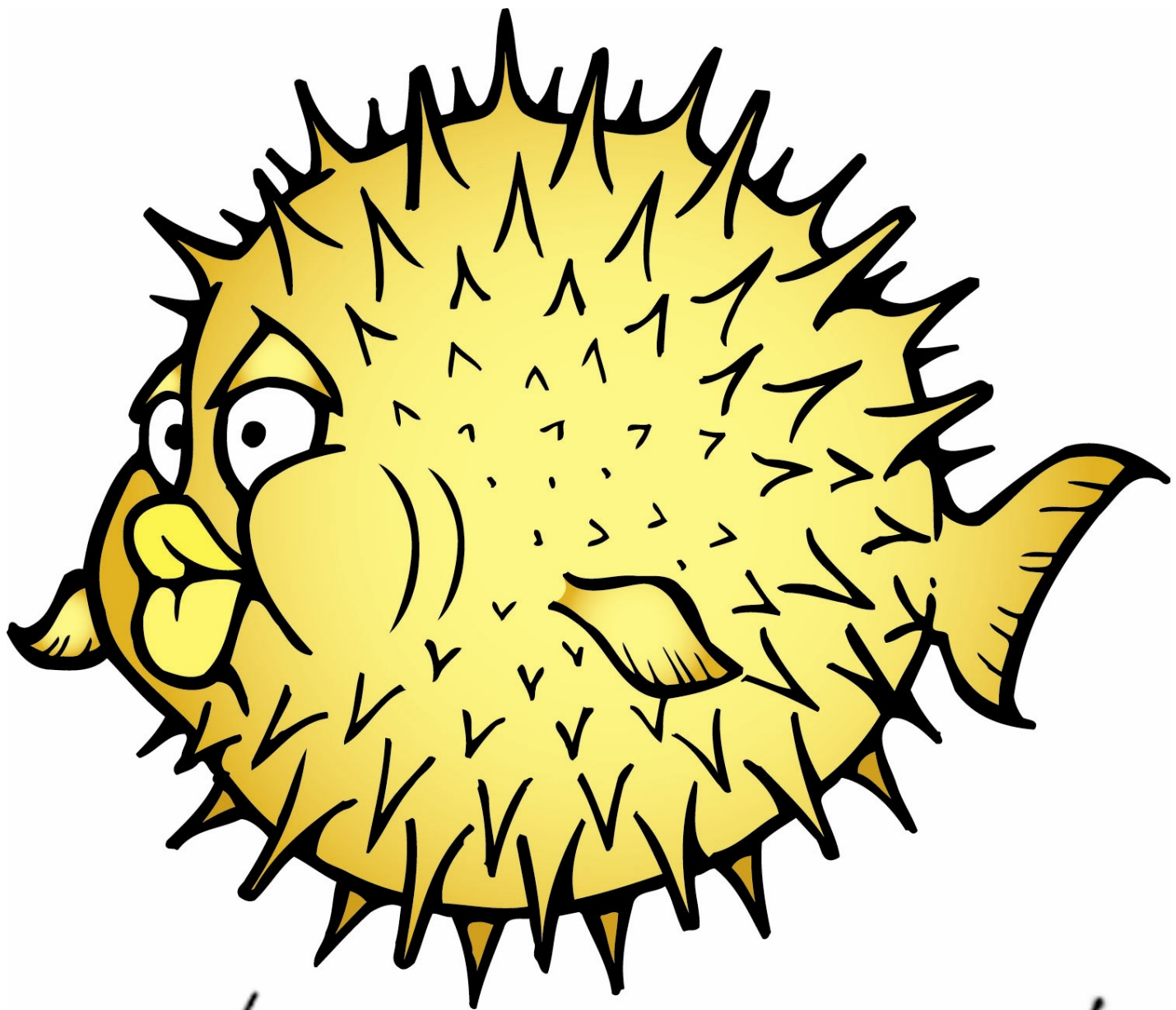
OpenBSD's pledge is well designed system security feature when properly implemented will help greatly to reduce the system attack surfaces by allowing for fine tuning of permitted system function calls and related operations.

***<http://man.openbsd.org/OpenBSD-6.0/man2/pledge.2>

****<https://gist.github.com/ligurio/f6114bd1df371047dd80ea9b8a55c104>

About the Author:

Albert Hui has been passionate about various operating systems and has been an OpenBSD enthusiast since 2003.



Strong crypto

Control Your Files Using Your Own Cloud With ownCloud

by Marcus Schmitt

If you search for cloud solutions within the internet, you are able to find hundreds of solution providers. But what happens with your private data? Who really has access to your data? Very restrictive file access settings within the cloud service doesn't guarantee that the cloud company won't copy or forward your private files. Furthermore, some providers simply scan your data to create a user profile in order to sell the profile to marketing organisations.

This is where ownCloud comes into the game. OwnCloud is an open source file storage and file sync solution. It provides access to files through via web interface, webDAV or sync client which makes it very easy to sync files from the file explorer to the cloud. Additionally, multiple add-ons can be installed within ownCloud to make the system even more useful. For example, there are plugins to create notes, a calendar app and an address book.

OwnCloud was first mentioned during Camp KDE 2010 by the founder Frank Karlitschek. A couple of months later, ownCloud 1.0 was announced, which included the web interface, WebDAV support and the plugin system as well as the option for notifications. During the next six years, Frank Karlitschek, Holger Dyrioff and Markus Rex added multiple features, created the desktop client, introduced the Security Bug Bounty program, and improved the software until the current available version: 9.1. Frank Karlitschek left ownCloud at the time ownCloud 9 was officially announced in order to create a new cloud project called NextCloud. There are many discussions about the reason for the new project but this should not be part of this article.

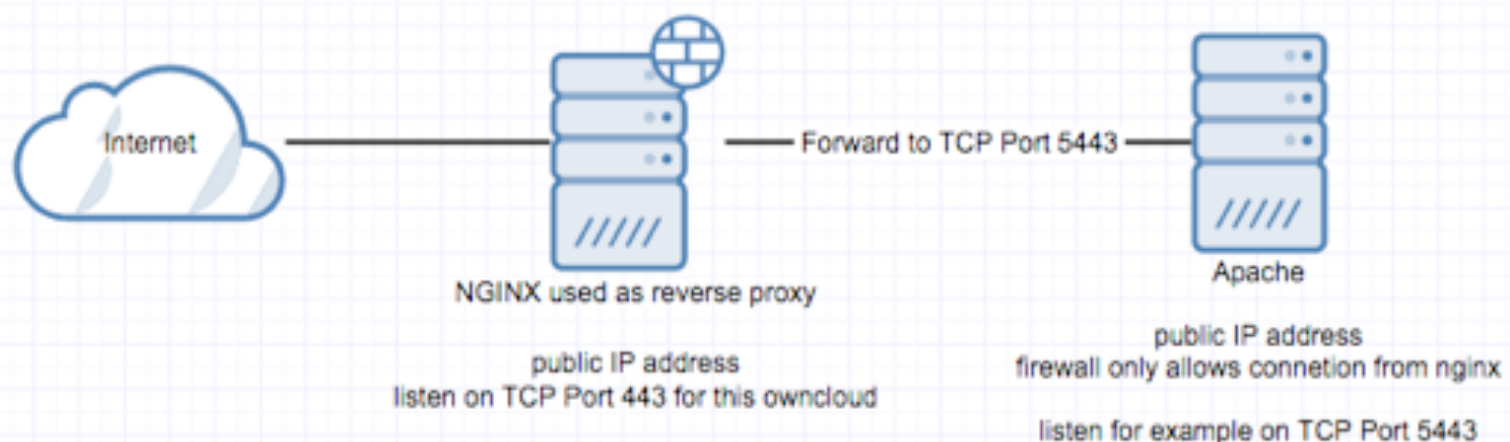
*

Like many other open-source projects, ownCloud provides a community and an enterprise version. The enterprise version includes features that the community version does not. For example, in the community edition, it is not possible to simple drag and drop the files from file explorer to the browser window. Furthermore, the community version does not provide access for guest users and it is not possible to integrate Microsoft Sharepoint or Windows Network Drives, or to use Single Sign On for Authentication. An overview of the differences between the editions can be found at <https://owncloud.com/community-or-enterprise/>.

BSD users should note that ownCloud has no official support for BSD so even the enterprise support will not be helped with BSD-related questions. However, many ownCloud problems encountered during the installation or the maintenance process can be solved by looking at the ownCloud community forum or reading through the extensive user and admin guides.

This article will show the necessary steps to install and maintain ownCloud on FreeBSD 11. Starting with installing the webserver, through configuring MySQL to set different parameters to harden the system.

In the first step, we have to install a web server to present the ownCloud website. For our environment, Apache in version 2.4 is just working fine. Later on nginx can additionally be installed as a reverse proxy to increase the speed for handling SSL/TLS connections. Furthermore, nginx as reverse proxy really helps if your root server does only have one IP address but you want to run multiple web pages on port 80/443. In this case, nginx forwards requests for different URLs to the apache, which listens on different internal ports. (ZEICHNUNG).



*

For now we only want to install Apache using the following command:

```
portmaster www/apache24
```

```
===>>> The following actions were performed:
```

```
Installation of databases/db5 (db5-5.3.28_6)
```

```
Installation of databases/gdbm (gdbm-1.12)
```

```
Installation of textproc/expat2 (expat-2.2.0)
```

```
Installation of devel/apr1 (apr-1.5.2.1.5.4_2)
```

```
Installation of devel/pcre (pcre-8.39)
```

```
Installation of textproc/libxml2 (libxml2-2.9.4)
```

```
Installation of www/apache24 (apache24-2.4.23_1)
```

In order to enable Apache 2.4 during boot, we simple use the command "`sysrc apache24_enable=yes`", which simply adds the command to `/etc/rc.conf`.

By default, OwnCloud uses MySQL in order to read and write different data used by ownCloud. Therefore, it is necessary to install some kind of database like MariaDB, PostgreSQL, MySQL or Oracle. (Oracle is only available for ownCloud Enterprise Edition). To install MySQL:

```
portmaster databases/mysql56-server
```

```
===>>> The following actions were performed:
```

```
Installation of archivers/liblz4 (liblz4-131)
```

```
(...)
```

```
Installation of databases/mysql56-client  
(mysql56-client-5.6.34)
```

```
Installation of databases/mysql56-server  
(mysql56-server-5.6.34)
```


In the same way, Apache was added to `/etc/rc.conf` we now add MySQL by using the following command:

```
sysrc mysql_enable=yes
```

In order to proceed with the next step, it is necessary to start MySQL with:

```
service mysql-server start
```

To configure MySQL, FreeBSD has a nice wizard. The wizard can be started by using the command:

```
/usr/local/bin/mysql_secure_installation
```

Recommendations from my side would be to set a long root password, remove the anonymous users, disallow root login remotely and remove the test database. Please keep in mind that for now, MySQL has no root password; this means you answer the first question simply by clicking return.

The next step is to install the Apache PHP module version 7.0 by using the command:

```
portmaster www/mod_php70
```

```
==>>> Installation of www/mod_php70 (mod_php70-7.0.13) complete
```

```
portmaster lang/php70 textproc/php70-ctype textproc/php70-dom
graphics/php70-gd converters/php70-iconv devel/php70-json textproc/
php70-xmlwriter textproc/php70-simplexml sysutils/php70-posix
archivers/php70-zip archivers/php70-zlib databases/php70-pdo_mysql
lang/php70 ftp/php70-curl sysutils/php70-fileinfo archivers/php70-bz2
security/php70-mcrypt security/php70-openssl converters/php70-
mbstring devel/oniguruma5 graphics/php70-exif security/php70-filter
security/php70-hash textproc/php70-wddx textproc/php70-xml www/php70-
session textproc/php70-xmlreader textproc/php70-xsl
```

```
===>>> The following actions were performed:

    Installation of lang/php70 (php70-7.0.13)

    (...)

    Installation of textproc/php70-xmlreader
    (php70-xmlreader-7.0.13)

    Installation of textproc/php70-xsl (php70-xsl-7.0.13)
```

Personally, I am not interested in connection to an LDAP server or SMB server, therefore I disable LDAP and SMB during the installation process.

Finally, we install ownCloud itself by using the following command:

```
portmaster www/owncloud

===>>> Installation of www/owncloud (owncloud-9.1.2) complete
```

After installing all the necessary components, it is necessary to configure the Apache server in order to display the installed ownCloud. Considering the fact that later on, we have to login using username and password, we will make sure that only login via HTTPS is possible.

In the first step, it is necessary to edit the file `httpd.conf` at `/usr/local/etc/apache24`.

The following file should be edited:

```
Replace "Listen 80" with "Listen 443"

Enable "LoadModule socache_shmcb_module
libexec/apache24/mod_socache_shmcb.so"

Enable "LoadModule ssl_module libexec/apache24/mod_ssl.so"

Disable "LoadModule status_module libexec/apache24/mod_status.so"

Disable "LoadModule autoindex_module
libexec/apache24/mod_autoindex.so"
```

```
Make sure that "LoadModule php7_module libexec/apache24/libphp7.so"
is enabled
```

```
Replace "ServerAdmin you@example.com" with "ServerAdmin <email-
address>"
```

```
Enable "Include etc/apache24/extra/http-ssl.conf"
```

```
Disable "Include etc/apache24/Includes/*.conf"
```

Add the following lines:

```
# Define to handle files with the ending .php as PHP Skript files

<FilesMatch "\.php$">

SetHandler application/x-httpd-php

</FilesMatch>


# Define to handle phps (php source files) by the PHP source-filter/
handler


<FilesMatch "\.phps$">

SetHandler application/x-httpd-php-source

</FilesMatch>


# Sets the list of resources to look for when the client requests an
index of the directory


<IfModule php7_module>

DirectoryIndex index.php index.html index.htm

AddType application/x-httpd-php .php

</IfModule>
```



```
# Disable HTTP Trace Method

TraceEnable off

# Disable showing the server version and only showing Apache in the
Server header

ServerTokens Prod

ServerSignature Off


# Control and modify HTTP request and response header

<IfModule mod_headers.c>

    Header always set Strict-Transport-Security "max-age=15768000; in-
cludeSubDomains"

</IfModule>
```

For the next step, it is necessary to edit `httpd-ssl.conf` in the `/usr/local/etc/apache24/extra` directory:

```
Disable "Listen 443"

Replace "DocumentRoot "/usr/local/www/apache24/data"" with "Documen-
tRoot "/usr/local/www/owncloud""

Replace "ServerName www.example.com:443" with "ServerName <your
fqdn>:443"

Replace "ServerAdmin you@example.com" with "ServerAdmin <email-
address>"

Replace "SSLCertificateFile "/usr/local/etc/apache24/server.crt""
with "SSLCertificateFile "<path to the server certificate""

Replace "SSLCertificateKeyFile "/usr/local/etc/apache24/server.key""
with "SSLCertificateKeyFile "path to the server private key""
```

A good way to generate a valid certificate is to use „Let’s Encrypt“.

For more information please have a look at „<https://certbot.eff.org/>“ where you can simply select your webserver and the operating system in order to get detailed instructions for your specific environment.

Add the following lines:

```
<Directory /usr/local/www/owncloud/>

    Options +FollowSymlinks

    AllowOverride All

    Require all granted


    SSLRenegBufferSize 888388608 (Comment: We can simply delete this
line because it was only used during different tests)

    <IfModule mod_dav.c>

        Dav off

    </IfModule>


    SetEnv HOME /usr/local/www/owncloud

    SetEnv HTTP_HOME /usr/local/www/owncloud

</Directory>
```

After changing multiple lines in both files, it is necessary to make sure that the certificates for the configured full qualified domain name is valid and located in the path configured at SSLCertificateFile and SSLCertificateKeyFile.

One of the last installation steps is to start Apache by the following command:

```
service apache24 start
```

Now ownCloud will be ready for the last step. So, just connect to the web interface and fill out the required fields. Keep in mind that there is no ownCloud database user yet, therefore use the database root user you created before, while we configured MySQL via wizard (`mysql_secure_installation`).

Now you should be able to use ownCloud without any problems.

Optional OwnCloud Hardening:

Based on your security requirements, there are some recommendations for hardening ownCloud.

1. Some of the default SSL enable ciphers in Apache are known as weak, therefore, there is a very helpful configuration wizard that shows cipher recommendations for different browsers at <https://mozilla.github.io/server-side-tls/ssl-config-generator/>. The disadvantage for this option is that older web clients might not be able to connect to the Apache.
2. In order to prevent security problems from another website on another virtual host, it is a good option to use one of the best builds in security features in FreeBSD, FreeBSD Jails.
3. When using Jails, you may have multiple jails each with a separate webserver that should all listen on port 80 or 443 but your server provider only provides you with one public IP address. One valid solution would be to use nginx reverse proxy to listen on port 443 but internally forward the requests to different ports on different jails.
4. One more religious option is to, on a periodic basis, fetch the country IP ranges in order to block the IP addresses outside the block requests from outside their own country. For sure, this does not increase the security itself but will reduce the number of global scans from outside their own country. The disadvantage is that some companies use internet service providers where the public IP address is used from different countries. In this case, the user will not be able to connect to the Apache.
5. In order to increase security, it is helpful to disable unneeded PHP functions. The recommendation would be to add the following line to `php.ini`:

```
disable_functions=exec,passthru,shell_exec,system,proc_open,popen,curl_multi_exec,parse_ini_file,show_source,eval,php_uname,getmyuid,leak,listen,diskfree,space,tmpfile,link,ignore_u
```

If there would be a module needed in future versions of ownCloud, you will see the following line in the `owncloud.log` `<module>` has been disabled for security reasons at

```
/usr/local/www/owncloud/<path>
```

Optional apps:

The main purpose of ownCloud is to store and share files but during the last year, ownCloud implemented a platform for different internal applications, which are really helpful.

1. **Bookmarks:** This app is helpful to create a bookmark collection for your favorite bookmarks. For sure, every browser has its own bookmarks store but using the app, it is possible to have your bookmarks in a central location for each browser you use.
2. **Calendar:** The calendar app does what the name says, it provides calendar functionality with the option to sync the data with mobile devices.
3. **Contacts:** The same as Calendar, the application is a wonderful option to have a summary of your contacts.
4. **Tasks:** Tasks is a very helpful app to manage your daily and weekly todos. The options are diverse. It starts from creating sub-todos and ends with priorities and tags for the different tasks. Perfect for organizing your day.
5. **Podcast:** The podcast app helps you to organize and play your favorite podcast within your browser by just adding the podcast URL.
6. **QuickNotes:** Just add important notes to nicely colored virtual notepads. Simple, but very helpful to quickly and effectively note information.

About the Author:

Marcus Schmitt is a Senior Network Engineer for network infrastructure and security in Germany. During multiple years of working at the Cisco TAC team, he focused on packet capture analysis as well as web security products. Presently, he is working as a senior network, infrastructure and security engineer, where he is additionally responsible for the internal company cloud solution.

He started with FreeBSD and OpenBSD around 15 years ago, when he considered both operating systems more interesting than different tested Linux derivatives.

FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**



Example of one-bit corruption

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.iXsystems.com/mini>



FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

<http://www.ixsystems.com/storage/freenas-certified-storage/>



Promote what you are doing with FreeBSD, and what you like and don't like.

Interview with Aryeh Friedman, CTO and Co-CEO of Friedman-Nixon-Wong Enterprises, LLC

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Aryeh, how have you been doing? Can you introduce yourself to our readers?

[Aryeh Friedman]: Since 2009, I've been co-owner of a small boutique software consulting firm, Friedman-Nixon-Wong Enterprises.

In the mid-1990s, along with John L. Sokol, I was the co-inventor of live Streaming Media, real-time streaming protocols (the best known modern version is RSVP), CDN's and SDN's (used to create the first streaming media network, which operated from 1994 to 2001), and single threaded web servers (thttpd is a modern version of our work). Concurrently, in the early-to-mid-1990s, I helped start the first generation of ISPs in Southern and Central California. (For details, see my LinkedIn profile and the links at the bottom of this interview.) I was also on an early incarnation of the IETF committee that led to the creation of ICANN.

In the late 1990s and early 2000s, I worked in eDemocracy, debates.com, one of the earliest social networking sites (although it wasn't called that back then). I also got interested in election security in this capacity and therefore have deep doubts about the integrity of the voting system in the 2016 Presidential election.

In the 2000s, I decided to go back to school and get my BS in Computer Science Education. (A side note; I do not agree with the "CS for all" current push in CS education, particularly in elementary school.) As I was graduating, the department chair introduced me to another student with an idea for a widget-based website builder.

While that company went nowhere, due to problems on the business side of the company, I did meet FNWE's current co-CEO and lead architect, Dee Nixon.

Working with Dee has allowed FNWE to do amazing work with a tiny team. For example, we maintain 800k lines of Java across our different projects with Dee and myself doing almost 100% of the work. I tend to be a highly creative but not super detail-oriented big-picture programmer, while Dee is excellent at getting all the i's dotted and all the t's crossed. We use a development methodology we call "sketch artist" from our strengths and weaknesses. Typically, I create a prototype and then Dee works on the most critical, detail-oriented aspects to ensure that they are rock solid in stability and robustness, while I work on aspects that are less detail-oriented but may require a wider range of technical knowledge. For example, I am the one that did the main thinking on how to do the transport and reporting of remote cardiac data for Specialized Medical, while Dee is the one who made it so it can be used in a high-security life critical environment.

[BSD Mag]: Can you tell us something about your company, Friedman-Nixon-Wong Enterprises, LLC?

[AF]: FNWE specializes in technical management and customized software based on our open source work. Our ideal client is a small to medium company that has healthy natural growth prospects (not the highly risky big bang growth that many VC's and other investors irrationally demand these days) and that requires the unique capabilities of the open source products we are currently developing.

Our open source work focuses mainly on secure cloud computing in life-critical applications. Our open-source projects include (1) a soon-forthcoming API/DB framework that can manage heterogeneous micro-services including a fully encrypted hierarchical database; (2) PetiteCloud, an IaaS platform designed primarily for use with FreeBSD as a host and bhyve as a hypervisor, with the robustness needed for small private clouds in non-data-center environments; and (3) thisTest, a Java unit-testing framework similar to JUnit but much faster and with a much lighter footprint.

Our main current custom/commercial software focus is medical labs, telemedicine, banking and other fields that require high security and the rock-solid stability demanded by life-critical applications. As a small consulting company, we've also done a variety of other projects, such as a poker odds calculator, a social media popularity algorithm, and various web development work requiring fully custom back ends and/or computationally sophisticated algorithms.

Our longest-term client is Specialized Medical, which does remote cardiac monitoring such as Holter and Cardiac Telemetry tests in real time. We developed and maintain a system that enables their clients (various private medical practices) to run multi-day cardiac monitoring tests on patients. This allows them to spot heart irregularities that a normal in-office EKG would not spot because it is too short in duration and does not follow the patient through their normal daily activity.

This combination of purpose-built open-source and custom software is particularly important in the industries we work in because, for example, HIPAA requires a formal (long-term)

“Business Associate” relationship to exist between the developer and the “covered entity” (the company that makes/manages the medical system).

When working on long term projects, we generally will agree to work with only one company in a given vertical market. For example, we are not available for work with other remote cardiac monitoring companies, although we are open for work with other kinds of medical labs, IoT, and tele-medicine applications.

Our advisers include a technical investment banker and a few OS developers. One of our advisers is Stanley P. Hanks, the CTO of Columbia Ventures, which, until recently, wholly-owned Hibernia Networks, which in turn owns the highest-capacity and lowest-latency transatlantic cable between Ireland and the US (NYC). Stanley was also the inventor of Internet VPN's while at MFS Datanet then went on to become the CTO of Enron Communications (not the part of Enron that was involved in the accounting scandal). At Enron he and his team co-invented CDN's independently from the team I was associated with at Sokol and Associates, described below, who also co-invented CDN's at the same time.

Another adviser of ours is John L. Sokol, who, besides being my boss at the early streaming video company DVBS in the mid-1990s, was a member of the team that originally put 386BSD up on UseNet for people to download, back around 1990 or so. FreeBSD 2.0.5 was forked from this late 1980s effort. The first time I had heard of 386BSD was when I met Bill Joltz in a Berkeley bookstore in 1986. I had already been using BSD since 42BSD at that point, since I grew up in Berkeley.

[BSD Mag]: Tell us something about your open source products. Which open source solutions do you use and why?

[AF]: Our main open-source projects have the eventual goal of making an IaaS/PaaS framework that can be used on the public Internet while meeting the end-to-end encryption requirements of HIPAA and other high-security standards. Preliminary steps toward that goal include (1) a small-scale IaaS framework, PetiteCloud, and (2) pApi, a hierarchical API/DB framework that allows full encryption of the DB, as well as management of other kinds of resources.

As far as we know, there is no other API/DB combo with a DB that encrypts entire files. All other solutions we know of use encrypted disks, SSL/TLS and/or encrypted DB fields, but NOT fully encrypted records or tables.

PetiteCloud, our IaaS framework, is now good enough for small-business in-house use but not yet good enough for data center use (e.g. it is not yet “lights out” capable, nor does it currently have the administrative interface needed for large scale deployment, nor does it yet have the security features we plan to add soon) – although it is already much more robust, in some ways, than the typical data-center IaaS platform.

For example, PetiteCloud can recover from power failures much more easily than, say, OpenStack, and we plan to keep it that way as we scale it up.

PetiteCloud is the only IaaS effort we know that is based on FreeBSD and bhyve as its main building blocks. The main design philosophy is to delegate as much as possible to the host OS, which puts us in contrast to more heavyweight IaaS's like OpenStack. We are currently working on making PetiteCloud fully HIPAA-compliant and data-center-ready.

We are also creating a hierarchical API/DB framework called pAPI, which will become the foundation of our PaaS: thinStorm. pAPI can manage heterogeneous collections of resources including, among other things, records and tables of the aforementioned fully encrypted hierarchical DB.

Another open-source project of ours that is more mature is thisTest, a Java unit-testing framework similar to JUnit but much faster and lighter weight.

Since paid work takes priority over our open source work, for obvious reasons, progress on PetiteCloud/thinStorm is not as fast as we wish. For that reason, we plan eventually to launch something like the FreeBSD Foundation around PetiteCloud/thinStorm (and pAPI), and in that way, enable faster progress without requiring FNWE to become a large firm (we want to stay relatively small).

We love the FreeBSD development model, in contrast to the Linux model. The FreeBSD model allows for more coherent and focused open-source work. Since we use the BSD license for all our open source work, we also welcome others to use it for their commercial products without having to pay us royalties.

[BSD Mag]: Is your solution designed for banking and healthcare mostly or can it be used in any industry?

[AF]: The next major version of PetiteCloud/thinStorm will be designed for any cloud computing applications that require true end-to-end encryption. The need for security is one of the reasons we chose FreeBSD over other OS's. The other main reason is the legendary stability of FreeBSD, both as a host and as a guest. PetiteCloud/thinStorm currently runs on both Linux and FreeBSD as both host and guest (using QEMU on Linux and bhyve on FreeBSD). We will soon be updating PetiteCloud to allow Windows guests to run under bhyve (it already runs fine under QEMU). We also plan to expand our hypervisor options to include VirtualBox.

As we are nearing the first sufficiently heterogeneous version of pAPI, we will be converting PetiteCloud over to it and making PetiteCloud truly end-to-end encrypted. This means it will be usable in any secure setting, not just medical and banking. We will then turn our focus to thinStorm to make the only open source PaaS designed for security from the ground up. It will run on hypervisors and not containers/jails, because the latter do not offer enough separation between the host and guest for the security features we want.

Another unique aspect of all our work is that, since we are not associated with any large hosting company and/or data center, we are designing PetiteCloud/thinStorm to be used outside of data centers (as well as, eventually, in them). For example, the OpenStack documentation describes power loss as “the worst possible disaster” that can happen to a clouded data center (largely due to using iSCSI instead of more fault tolerant network file systems like NFS backed by a ZFS file server). Since we run PetiteCloud in our non-purpose-built office, it routinely loses power due to things like one of us kicking the power strip while cleaning the room. The only recovery needed, typically, is just hooking the power back up. OpenStack, on the other hand, will brick up if it loses power for as little as one second.

All the above will make PetiteCloud an ideal private/hybrid cloud needed for high security operations in small and medium businesses. For example, once all the security features are added, it will be ideal for a small bank, law firm, medical clinic/small hospital, etc. We estimate that will enable it to be used by the 49% of the computing world that requires security better than what can be offered with OpenStack or with commercial cloud providers (without contracting with them for a private cloud). Since we plan to use PetiteCloud/thinStorm to support HIPAA compliant custom electronic medical records systems, it will meet PCI-DSS also if properly secured physically and run on a private cloud.

[BSD Mag]: Do you have your favorite open source system?

[AF]: There is not a single system I like the best, but the combination of tools listed below give us an amazing foundation to build our open source and custom systems on.

I am a FreeBSD fanatic and have used it since 2.0.5, so I would say that FreeBSD is by far my favorite open source platform. OpenJDK is a close second, though, because Java is uniquely well suited to the type of development we do. We like Java because it has the software engineering characteristics that allow us to avoid – or, if necessary, quickly debug -- life-threatening bugs in a life-critical application, without compromising on the security (which is a legal requirement for our clients).

Also, in the interests of keeping bugs to a minimum, we believe that change management is of critical importance in large systems. By change management, we mean not just version control but also atomic change sets, with controlled access to the baseline/repository. For that reason, our preferred development environment is a combination of devel/aegis (which I am the port maintainer of) and devel/cook (for which I've written a tutorial). Both these tools were developed by Peter Miller and are still, as far as I am aware, the only tools that satisfy all of his three laws of change management. His laws are:

1. Without controlled access to the baseline, the number of interactions within a development team is $O(n!)$, where n is the number of developers and/or the number of files in the source tree-- whichever is larger. With controlled access to the baseline, it can be reduced to near $O(n)$.

2. The baseline MUST always be in working order.
3. The software build/construction process can be reduced to a directed acyclical graph (DAG), as described in his paper “Recursive Make Considered Harmful”.

The first law addresses the main reason for change management systems, namely source-code control. When you have too many people simultaneously interacting with the code, unless you make sure each is working on local copies until they are ready to merge them back into the master copy, they will constantly step on each others' feet.

Then, there is the second law that only Aegis enforces. A good change management system should make it difficult to check in buggy, non-working code and integrate it into the baseline. This means atomic checking in of change sets, in contrast to the far less robust check-in procedures of git and most other version control systems. I have been a strong advocate of the FreeBSD base system switching over to atomic change sets vs. the git model. If it had, my estimate is that 11-RELEASE could easily have been on time instead of being almost six weeks late.

[BSD Mag]: You have been participating in couple of projects and volunteer activities. Have you ever been a part of open source community? Or is it security you are interested in more than open source?

[AF]: I have been associated with several open source efforts and we plan to use that experience to build a strong non-profit organization to handle the care and feeding of PetiteCloud/thinStorm in the long run. As stated above, I am also the maintainer of several FreeBSD ports. My main areas of interest are cloud computing and security currently, but I am also interested in other types of open source projects as well.

During the early and mid-2000s, I was one of the founders of the now-defunct Software Developers Cooperative (SDC) that sought to create a set of licenses that would not need dual licensing to use open source for commercial purposes. At the time, I had a false understanding of the BSD license; I thought it, like some other open-source licenses, forbade commercial use. Once this misunderstanding was resolved, I dropped out of SDC and started using the BSD license exclusively for my open source work. Around this time, I wrote several blogs that examined the problems GPL created for developers who do not get subsidized by their employers/schools for their open source work. The primary issue here is that, while the BSD license is both free beer and intellectual freedom, GPL is only free beer unless you happen to have a well heeled employer behind you instead of making a living from your own work.

For this reason, the model we will be using with PetiteCloud/thinStorm is a fully free and open-source core with commercial or FOSS add-ons made by competing groups. The core, though, will be maintained by a single organization. The closest model is that of the FreeBSD base system

This is a specific example of a larger small business/open source business model we envision, called neo-Jeffersonianism, which is intended to enable small companies to compete effectively as clusters against even the largest and most entrenched competitors without losing their individual identities. We believe that, if properly structured, neo-Jeffersonianism could be one of the few scalable sustainable economic growth models we know of.

Thomas Jefferson wrote: “I hope we shall crush in its birth the aristocracy of our moneyed corporations which dare already to challenge our government to a trial by strength, and bid defiance to the laws of our country.” We don't advocate getting rid of large corporations entirely, since there are many economic activities that can be done only by large corporations. But we believe that the power of large corporations needs to be counter-balanced by organized clusters of small businesses. Jefferson envisioned a world in which the majority of families owned small farms. That particular goal is obviously outdated in today's world of mechanized agriculture, which has freed up the vast majority of people to do all manner of other things besides farming, but the next best thing, consistent with Jefferson's goal of limiting the power of big corporations, is to encourage the creation of organized business clusters that can enable small businesses to survive and thrive without requiring huge monetary investments.

[BSD Mag]: You also have a patent! Tell us something about it.

[AF]: A small disclaimer: I am an un-named inventor on the patent, because I left the company (Sokol and Associates) before the paperwork was complete and thus only John L. Sokol's name appears on the patent itself, but he has given me credit in the original documentation and elsewhere.

The patent is for a single threaded web server called AfterBurner, currently posted on SourceForge under the BSD license. The idea is that, for static content (i.e. stuff that does not need a backend), the maximum hit capacity of the web server can be cranked up much higher than with a threaded web server such as Apache or Tomcat. There are several current web servers based on this design now, with the best known being thttpd.

In testing the prototype of AfterBurner in 1994 and 1995, we were able to support, on a single Pentium-90, almost the entire load that Yahoo! was reporting for their entire site, yet the CPU was still 90% idle. The same machine then maxed out four 1,000 Mb/s Ethernet NIC's and was still running only at 15% capacity. As far I know, AfterBurner still holds the raw performance record for any web server.

John Sokol and Terry Lambart later adapted the same model to create the first kernel queues implementation for BSD. My understanding is that a variant of this model is still used in the FreeBSD kernel.

The other items I mentioned above as being the co-inventor of were judged to not be patentable

by Sokol & Associates' IP attorney. I guess I will have to live with bragging rights only on them. (See list of links for details.)

[BSD Mag]: What is Rent-a-CTO? Sounds like “Rent a Chief Technology Officer” ;)

[AF]: One of the largest misconceptions many non-technical founders have is what exactly the role and function of a Chief Technical Officer is. The standard assumption is that it is some kind of super techie who can jump over tall buildings in a single leap and can write code by just thinking about it (no typing needed). The reality is that being a CTO is largely a business position rather than a technical position, although it does require wide-ranging technical knowledge and experience.

The main jobs of a CTO are to develop a coherent technical strategy, explain the technical aspects of the company to the non-technical stakeholders, and, in a start up, assemble the technical team that will do the R&D and then continued support of the company's products and services.

This means that most small and startup companies don't need a CTO except when they are making a pivot from one phase of their life to another, and/or when they grow. The rest of the time they don't need a full time person in this role. What many small companies and startups do, therefore, is to have one full-time person in the role of both CTO and lead developer. However, a wider range of technical knowledge can be brought to bear if the company has both a full-time lead developer and a very experienced consultant acting as a part-time CTO. FNWE can provide either development services or part-time CTO services, as needed.

[BSD Mag]: What are the challenges your company is facing at the moment?

[AF]: The main challenge we face is how to balance the demands of our work for clients vs. our open source work. We have a policy of not billing clients for open source work, even when their projects benefit from it.

For this reason, we want to move to a non-profit foundation model for PetiteCloud/thinStorm. We would, of course, be pleased if some of our clients, as well non-clients, made donations of their time and/or money to the foundation. Sustaining members would have a say in the project's future direction without having to pay, individually, more than a small fraction of the cost. The main goal of the foundation will be to allow a wider ownership of our open-source projects beyond just FNWE, so we can get paid for at least some of our open-source work while continuing our policy of not billing our clients for it. We do hope, at some point in the future, that the foundation can support one or more full-time project developers, since that would speed up development significantly.

Until then, our largest challenge is balancing the demand for immediate paid work with the long

term investment needed to make PetiteCloud/thinStorm a reality.

Fortunately, since we have no external pressure from investors or other non-clients, we can allow PetiteCloud/thinStorm to evolve as needed without the need to slap on features that may be ill advised from a technical standpoint but the marketing department demanded them.

[BSD Mag]: Any plans for the future?

[AF]: From the personal stand point, I would love to find enough time to go back to school to get my PhD, but from what I have learned from talking to others in my position, this would be largely just a piece of paper, given my background.

From the technical stand point, I would eventually like to leverage PetiteCloud/thinStorm into a fully distributed OS design I have been working on for about 10 years now. With that OS, I plan to move as far past cloud computing as cloud computing has moved past traditional IT.

For FNWE, we plan to continue to grow organically so that we can fund the development of this OS without the need for outside investors. I want to see neo-Jeffersonianism become a much more widely used business model because it allows for the average person to start a small firm that will grow organically while at the same time returning the economy to a more balanced ecosystem between small and large companies.

Micro-economists, like my father, have long known from experimental and other real world studies that the more times the same dollar circulates within a semi-closed economy, such as the one found in pre-civil rights black neighborhoods or currently found in many immigrant and religious minority communities, it does that much more work. For example, in the average pre-civil rights black neighborhood, a dollar would circulate within the community six times before leaving the community for the larger economy. This means that dollar essentially does six times more work than the same dollar in a mass of atomized individual actors. In the atomized case, there is much more overhead (e.g. advertising expenses and longer times required to build a suitable network of business connections). In the atomized case, there is an algorithm (known as Simple Marx) that shows that eventually, everything else being equal, one actor will end up holding all the gold.

One of the main goals of neo-Jeffersonianism is to encourage the creation of many different, but interlocking, semi-closed economies that can easily capture dollars in a manner that allows the money to circulate multiple times within a relatively small community before leaving it.

The communities do not have to be racial, ethnic or religious. Neo-Jeffersonianism can be structured around small business clusters. A business cluster is a group of companies that are complementary in some way (exactly how is up to the cluster members).

The eventual structure for PetiteCloud/thinStorm development and deployment is one type of such a cluster. In the case of PetiteCloud/thinStorm, we plan to build a group of companies each

of which, while deriving some or all of its revenue from PetiteCloud/thinStorm-related products and services, is not large enough by itself to shoulder more than a small fraction of the R&D, general marketing/PR, etc. the project requires. Thus, there is a common good that all cluster members have access to, but at the same time they are free to develop their own strategies about how to maximize their own potential. For readers who understand economic theory, think of it as collective ownership and oversight of the commons in order to avoid the tragedy of the commons.

[BSD Mag]: Do you have any piece of advice for our readers?

[AF]: The pace and bustle of tech in general and Silicon Valley specifically should not defocus you from your core goals and vision. The extremely rapid pace of progress in computer technology over the past 50 years has been due mainly to Moore's Law, not to advances in computer science or software engineering, which have moved at a glacial pace compared to sheer capacity increases due to Moore's Law. Since Moore's Law is near its physical limits, the pace of progress on the hardware front will soon slow down. Hence, we as an industry will need to focus more on how to make our software more efficient in terms of performance, resource demands, and developer time/effort.

Sadly, as Fred Brooks stated, there are no silver bullets to these problems; hence, we need to continue honest and open efforts to improve. We should not settle for fake silver bullets like Agile Development and its spinoffs -- although, at the same time, pure waterfall is completely dead if it was ever alive in the first place. Ditto for design patterns: Model-view-controller, for example, is excellent for many applications but not appropriate for everything. Only with experience can programmers learn which methodologies are appropriate for which kinds of projects, or find the happy middle ground between the latest fad and older software engineering practices.

On the social level, tech can improve life for everyone, but only if it becomes truly and democratically open to all. Silicon Valley has created a bubble around itself and fooled itself into thinking that it is in fact diverse and inclusive. Nothing can be further from the truth. NYC is much more diverse in its point of view and startup culture, but even in NYC, due to the investment focus of most companies, their values get warped.

So I guess one piece of advice I would give is that tech and CS have a lot to offer the world but as long we allow unrealistically rapid growth/profit expectations to be the primary driver, they will fall short. The only way to avoid excessive focus on rapid growth is to return to a small business focus.

Also, if you are just starting out in the field, you should focus on building a solid foundation for a decades long career instead of just going for the "hot" tech of the moment. Like the world of high fashion, the hottest new tech is often impractical in the real world, and often has little value beyond being flashy and "cool". One should focus, instead, on finding the best ways to solve real problems for real people.

For more information, see:

Real-time Transport Protocols (S-Pack and ECIP): <http://ecip.org/> (see history page: <http://ecip.org/history.html>)

Streaming media (done for DVBS, Inc.):
<https://www.quora.com/Who-invented-streaming-video>

AfterBurner/Web server/patent:
<http://johnsokol.blogspot.com/2006/11/afterburner-web-server.html>

<https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2000041455>

CDN/SDN (Race and SDSN): <http://www2.videotechnology.com/old1005.html>

Stanley P. Hanks and VPNs: <https://www.rfc-editor.org/rfc/pdf/rfc/rfc2784.txt.pdf>

John L. Sokol and 386BSD: <http://www.dnull.com/bsd/others/17.txt>

Specialized Medical: <http://www.specialized-med.com/>

Columbia Ventures: <http://colventures.com/>

Hibernia Networks: <http://www.hiberniaatlantic.com/>

Aegis/cook: <http://aegis.sf.net>

Recursive Make Considered Harmful: <http://aegis.sourceforge.net/auug97.pdf>

PetiteCloud: <http://www.petitecloud.org>

FNWE: <http://www.fnwe.net> (under construction)

Cook tutorial: <http://ibiblio.org/pub/Linux/devel/make/cook-2.25.tut.pdf>

Aegis port (FreeBSD):
<https://www.freebsd.org/cgi/ports.cgi?query=aryeh.friedman%40gmail.com&stype=maintainer&sektion=all>

About the Aryeh:

Aryeh M. Friedman is Co-CEO/CTO of Friedman-Nixon-Wong Enterprises (FNWE), a small boutique consulting firm specializing in mission-critical secure cloud computing. He draws on over 25 years of industry experience including being a pioneer in the technologies that enable cloud computing. FNWE offers development and technical management services. Mr. Friedman currently lives in New York City but grew up in and around Silicon Valley before leaving it to find wider technical opportunities after high school. You can contact him at aryeh.friedman@gmail.com or via his linked-in profile at <https://www.linkedin.com/in/friedmanaryeh> .

DEVOPS WITH CHEF ON FREEBSD

General description:

This training class teaches the tools, best practices and skills to automate your FreeBSD servers. Training will be loaded with practical real world tools and techniques. This training will send you back to work with immediately useful hands on experience to implement Devops in your IT projects.

Course launch date: **1st of December 2016, Self-Paced.**

Duration: 18 hrs.

What will you learn?

- Learn what Devops is and its importance.
- Learn to leverage infrastructure automation using the leading configuration management tool: Chef.
- How it's changing the industry.
- Transform IT from an unpredictable environment to a stable, repeatable and scalable environment.
- Integrating configuration tools into the IT workflow.

What skills will you gain?

- Configure development workstation.
- Understand the Chef architecture.
- Understanding different resources and automation.

Module 1: Introduction to Devops and Chef.

Module 2: Understanding Chef resources.

Module 3: Setup the workstation as a local development environment.

Module 4: Bootstrap a node with Chef server.

Module 5: Get started with writing your first cookbook

Module 6: Recipes, Attributes, Metadata, Templates, etc.

Module 7: Roles, Environment, Search, etc.

Module 8: Using syntax and linting tools like: Foodcritic and Rubocop

Module 9: Writing unit test and integration tests.

There is only 50 seats available, so hurry up, book your seat as soon as possible.

You can find a full description of the course here:

<https://bsdmag.org/product/w05-devops-chef-freebsd/>

If you will have any questions regarding the course, don't hesitate to contact us at

marta.ziemianowicz@bsdmag.org

Why are simple matters of communication often so complicated? Lack of esprit de corps? No co-operation? You may have a psychologically toxic individual on your team, or worse still, encountered a professional culture that is poisonous.

by Rob Somerville

You can always tell when the natural dynamic of a team is “off”. Rather than experiencing a positive “Can do” attitude, a sense of synergy and unity, there appears to be an emotional elephant in the room. Certain subjects are taboo, and rather than “the whole being greater than the sum of its parts”, there is a mysterious black hole where any positive energy, ideas and creativity are sucked out of you.

The problem is that as human beings, we suffer from two traits that make identifying these issues difficult when we first encounter a team, a project or an organisation. Firstly, we want to make a good impression, and err on the side of caution when judging others’ idiosyncrasies. Secondly, we look at the world subjectively, expecting others to play by the same rules and ethics that we personally would. The latter is the cause of the most pain when encountering a toxic environment, for try as we may, we cannot comprehend how anyone could behave in such a manner. We have been blind-sided, betrayed, abandoned to forces that we cannot understand. Often, once we become an accepted part of the group, there is an open secret as to the source of the rot. The resulting disconnect causes stress, frustration, anger and if left un-

resolved leads to disillusionment and a tiredness in the bones that forces the victim to distance themselves – either temporarily or permanently – from the issue.

I have been privileged to encounter the best and worst environments and individuals in my 40 year career so far. The best, it goes without saying, is a joy. The worst, on the other hand, leaves little opportunity other than for personal growth and a deeper understanding of the darker side of human nature and sometimes the sheer stupidity and paralysis that can accompany group-think. While it is easy to rapidly identify a particular individual as being the lowest common denominator (there is no “I” in team), if the culture itself is toxic, it may take a lot longer to understand that the system itself is at fault. The individual is then faced with the difficult question – Can I best change this organisation from inside or outside? This assumes that the individual is on the side of the angels, and is not so arrogant as to think they have all the answers. Humility is a key factor here, as well as a strong dose of self knowledge, provided we want to be part of the solution rather than part of the problem, of course. The toxic individual is frequently either so wrapped in their own self

belief or deeply ignorant as to the damage they are doing. The worst examples are so convinced of being right, they are proud of their ability to disrupt even the most efficient and stable team. They are quite willing to openly admit in public their anarchic and rebellious tendencies. While it is essential that there is some sand in the oyster, this level of irritation is almost pathological in its fervour, causing trouble for the sheer sake of it. An emotional or intellectual troll, if you like.

So how can we deal with such difficult situations maturely and efficiently? Firstly, we need to have a good degree of self knowledge and understanding. Until we realise that we are human, make mistakes, and often get the wrong end of the stick, we will not develop a sufficient barrier of empathy that will allow us to compassionately deal with the mistakes of others. It is not until we are protected by this “emotional airbag” that we can utilise our personal alert system – and realise we have just been involved in a major collision. Secondly, we need to accept, sadly, that there are individuals out there that do not have our best interests at heart – no matter how genuine, mature or educated we may be. Sometimes, we need to accept the fact that some people or organisations are just so broken that the only sensible option is to run in the opposite direction as fast as your legs can carry you.

The best indicators in this area are relationship, honesty and trust. The less we have of these factors, the more likely we will be in the danger zone. Of course, there is always the possibility that while everything is OK in the short term, we will encounter difficulties fur-

ther down the line. However, without these attributes, there is very little we can do as individuals to redress any problems other than an appeal to authority. If the situation is really poisonous, this can be truly counter-productive and can result in a “Them and Us” scenario, with long-standing grievances raising their ugly head further down the road.

The great thing about the modern technical environment is our ability to document and log everything. The old saying “A verbal agreement is only worth the paper it is written on” is especially true in toxic situations. Provided a decent audit trail that cannot be tampered with is present, we have a strong evidence base to support our position. Sadly, one of the lessons I have learned is you need to be guided by your intuition on this, as it would be churlish to document everything as a matter of course. If you prefer to conduct your relationships on the basis of a handshake or a verbal agreement, this is especially pertinent. If I had a pound for every time my alarm bells rang and I didn't get a commitment in writing for the goalposts to be moved at a later date to my disadvantage, I would be quite rich. I struggle with this, as I far prefer to be an optimist when dealing with people, but realistically I only have myself to blame as I could sense that something was “off”. Hindsight, as always, has 20 20 vision.

At the end of the day, only you can tell if the boundaries have been crossed from acceptable to toxic behaviour, be that on an individual or wider basis. The important thing is not to suffer in silence, letting the poison seep into your veins and destroying your idealism in the

process. While it is important to take a balanced view in light of other people, we can always take steps to protect not only ourselves, but the group as well. At the same time, it is essential to have a sense of humour, and the willingness to say sorry if need be. Caution – human beings at work.